宝典丛书 100万

MATLAB 科学计算与可视化仿真

宝典

8年MATLAB使用经验的总结。

全程实例解说MATLAB在数值仿真中的应用。

配套光盘中代码+视频多媒体教学，大幅提高学习效率。

MATLAB中文论坛（www.iLoveMatlab.cn）强烈推荐本书。

刘正君 编著



電子工業出版社
Publishing House of Electronics Industry
http://www.phei.com.cn

MATLAB
科学计算与可视化仿真宝典

本书重点剖析MATLAB R2008a版本的最新功能与特性，结合多个专业问题讲解MATLAB数值计算及数据的可视化功能，并全程实例解说MATLAB数值计算技术，目标更明确。本书点面兼顾，目录分类细致而科学，方便不同类型读者的快速查阅。配套光盘可以免去读者烦琐的代码输入工作，提高学习效率。

宝典丛书

- Excel 2007 VBA高级编程宝典
- Excel 2007 公式、函数与图表宝典
- 中文版Office 2007宝典
- 中文版Access 2007宝典
- 中文版Excel 2007宝典
- Office 2007应用技巧宝典
- Windows Vista宝典
- Maya 8.5宝典
- 3ds Max/VRay效果图高级渲染宝典
- 中文版Dreamweaver CS3网页制作宝典
- Photoshop图像合成高级技法宝典
- Photoshop CS3图层、通道、蒙版深度剖析宝典
- Photoshop CS3平面广告创意特效宝典
- 中文版AutoCAD 2009宝典
- Eclipse，Struts，Hibernate，Spring集成开发宝典
- Visual C# 2008宝典
- ASP.NET 3.5宝典
- Spring 2.0宝典
- PHP 5+MySQL 5 Web应用开发宝典
- 基于J2EE的Ajax宝典
- Fedora Linux基础应用与配置管理宝典
- MATLAB宝典
- Pro/ENGINEER野火版4.0宝典
- Pro/ENGINEER野火版4.0实例宝典
- SolidWorks 2008宝典
- Oracle 11g宝典
- Oracle 11g数据管理与优化宝典
- …………







责任编辑：于 兰
责任美编：秦 靖

本书贴有激光防伪标志，凡没有防伪标志者，属盗版图书

定价：89.00元（含光盘一张）

宝 典 丛 书

MATLAB 科学计算与可视化仿真宝典

刘正君　编著

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内 容 简 介

本书共27章，分为4个部分，详细讲解MATLAB的计算和数据表现功能，介绍利用MATLAB对科学问题进行计算与仿真，并针对部分专业问题，给出利用MATLAB进行模拟程序和仿真结果。

本书第1部分包括第1~6章，介绍MATLAB基本知识：数据类型、向量与矩阵的定义、表达式、程序结构与优化、文件处理。同时，还给出一些实用经验促进读者更好地利用该软件。第2部分包括第7~15章，详细介绍基本科学问题的求解方法，如线性方程组、超越方程、数据拟合与插值、最值问题、随机数、微分方程组、积分运算、数学变换、特殊函数等。第3部分包括第16~18章，具体介绍二维和三维图形的绘制、用户图形界面设计等。第4部分包括第19~27章，具体介绍混沌、分形、元胞自动机、光学现象、机械运动、常用算法等方面的编程知识。

本书写作结构清晰，图形与程序结合，实例丰富，实用性强。通过实例详细地对实际问题进行了剖析并讲解如何用程序实现。

本书适合有一定数学基础和专业知识、对MATLAB不熟悉的人士，以及利用MATLAB进行数值模拟研究的人员参考使用，也可作为学校或研究机构及企业中利用MATLAB进行数值计算的教程，或自学MATLAB的读者的参考用书。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，侵权必究。

图书在版编目（CIP）数据

MATLAB科学计算与可视化仿真宝典 / 刘正君编著.—北京：电子工业出版社，2009.4
（宝典丛书）
SBN 978-7-121-08579-6

Ⅰ.M… Ⅱ.刘… Ⅲ.计算机辅助计算－软件包，MATLAB Ⅳ.TP391.75

中国版本图书馆CIP数据核字（2009）第044866号

责任编辑：于　兰
印　　刷：北京东光印刷厂
装　　订：三河市皇庄路通装订厂
出版发行：电子工业出版社
　　　　　北京市海淀区万寿路173信箱　　邮编：100036
开　　本：787×1092　1/16　　印张：42.75　　字数：1231
印　　次：2009年4月第1次印刷
定　　价：89.00元（含光盘一张）

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至zlts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。

服务热线：（010）88258888。

序

美国 The MathWorks 公司推出的 MATLAB 语言一直是国际科学界应用和影响最广泛的三大计算机数学语言之一。MATLAB 语言有其他基础程序语言无法比拟的优势和适用面。近 10 年来，随着 MATLAB 语言和 Simulink 仿真环境在各学科领域中日益广泛的应用，我国的科研工作者和教育工作者也逐渐将 MATLAB 和 Simulink 语言作为首选的计算工具。

随着科技的不断进步和发展，科研工作者在从事研究的过程中深刻体会到仿真验证的重要性以及数据交换运算的优越性。无疑 MATLAB 在这方面具有天生的优势，我们可以认为只要能用数据描述的对象必然可以使用 MATLAB 进行分析和研究，并且一旦熟悉相关工具箱函数以后，一系列的复杂运算和让人头痛的编程工作已经不能再困扰我们。

正是由于 MATLAB 所具有的强大运算功能和广泛的适用性，使得 MATLAB 以极快的速度在扩展自己的应用功能。MathWorks 公司近几年不断推出新版本软件，每个新版本都有令人惊叹的新工具和新功能，这又使得更多人对这款软件趋之若鹜。然而面对具有如此超强功能的一款“巨型”软件，即便是从事多年 MATLAB 研究的专家也只能对其庞大的功能“望洋兴叹”，对初学者来说，要掌握这门工具语言几乎成为一项“不可能完成的任务”。

市面上虽然充斥着各种介绍 MATLAB 语言的书籍，但熟悉 MATLAB 语言的人都知道，专业书籍大多只能对某个方向或领域的相关内容做出较为详细的阐述，当遇到基础运算和操作的问题，以及一些边缘算法时，很多有多年经验的研究人员也不得不查阅基础书籍或者求助于 MATLAB 的帮助文档。但是我们都知道 MATLAB 帮助文档是英文的，这对于很多国内的学习者来说是一大“拦路虎”。而那些基础的书籍基本上只能是对基础函数做一些概念性的介绍，不能全面细致地帮助我们解决实际问题。

刘正君博士组织编写的《MATLAB 科学计算与可视化仿真宝典》，正是可以帮助您解决各种研究和应用中实际问题的最佳参考资料。作者多年从事 MATLAB 的应用研究工作，在图像加密、光学变换、光束整形、混沌、分形、元胞自动机等领域有着丰富的经验，同时在 MATLAB 技术论坛长期解决各类学习者在使用 MATLAB 技术时遇到的各种问题，非常了解广大使用者在应用时所面临的各种难题。

《MATLAB 科学计算与可视化仿真宝典》一书，从最基础的 MATLAB R2008a 版本安装方法和一些基本操作知识入手，全面详细地介绍了数值及其科学计算的基础知识、数据可视化仿真操作及科学编程等内容。全书每个知识点都配以实例解说相关功能和操作，是专业学习和研究工作者值得参考的经典书籍，同时该书也非常适合初学者作为入门引导。在此，MATLAB 中文论坛向广大读者隆重推荐此书。

张延亮

MATLAB 中文论坛创始人 (www.iLoveMatlab.cn)

于新加坡南洋理工大学

作 者 简 介



刘正君，毕业于哈尔滨工业大学光学专业，理学博士。主要研究方向包括图像加密、光学变换、光束整形、混沌、分形、元胞自动机等。MATLAB 论坛技术版的版主，有大量 MATLAB 程序编写经验，以及使用 MATLAB 技术近 8 年时间的积累，尤其对数值计算和可视化等方面有深入研究。

前　　言

MATLAB R2008a 在 2008 年 3 月正式发布，该版本对很多工具箱中的函数进行了更新和补充。在该版本中，解决了 MATLAB 几个长期以来固有的弊端，同时加入一些重要的功能，如完全实现面向对象编程、支持 Handle 类型、引入命名空间的管理等。这次更新是非常富有战略眼光的，也代表了 MATLAB 的一次历史性转型。

MATLAB 是一款适合不同专业的人士解决问题的软件，其最大特色在于其根据需要不断扩充工具箱。即使不是专业软件开发人员，也可以调用 MATLAB 中的工具箱进行计算，再借助通用数据类型交换数据，从而利用 MATLAB 强大的科学计算功能。

随着科学的发展，使用数值仿真来验证定理或者结论的方式已经成为一种重要的手段。它具有快速、节省成本及灵活多变等特点。而 MATLAB 已经在数值仿真任务中占有统治地位。同时它的版本每年更新两次，及时扩充自身的功能，应用专业领域广泛。这一点是很多同类软件无法比拟的。

MATLAB 软件是基于 C 语言和 FORTRAN 语言编写的。但是 MATLAB 对很多功能都已经进行了函数化：一个复杂的计算任务，在 MATLAB 中常常用一条语句就可以实现。同时对于初学者，该软件很容易入门。伴随着使用者对问题研究的深入，可以积累 MATLAB 程序，常用的程序段可以写成函数文件的形式，有一定数目的程序文件之后，就可以建立自己的“工具箱”。本书编写从简单问题入手，逐步扩展到专业问题的求解。本书对于 MATLAB 函数的介绍采用统一格式进行，同时给出大量实例帮助读者理解函数的功能。

熟练应用 MATLAB 需要一定时间，读者在安装好 R2008a 版本后可以运行本书给出的程序，通过修改参数查看输出变化以了解 MATLAB 函数的功能。另外，需要注意的一点是本书介绍的少部分函数属于 MATLAB R2008a 特有的，较低版本因缺少相应函数会出现错误提示。为了节省读者输入程序的时间，本书的配套光盘中提供了 MATLAB 代码，为程序代码实现高效率复用提供便利条件。

主要内容

本书全面地讲解 MATLAB 数值计算和数据可视化仿真方面的功能，全书分为 4 个部分，共 27 章。具体内容如下所示。

第 1 部分　基础篇　包括第 1～6 章。本部分首先介绍 MATLAB R2008a 版本的安装方法和一些基本操作知识。对于新手来说，了解基本操作是非常重要的。总体来说，该版本保持了以前版本的大部分功能，熟悉以前版本的读者可以很快习惯这个版本的使用。接下来介绍数据类型、向量和矩阵的使用，它们是编程的基础。随后介绍不同类型的表达式，它们是进行数值计算的纽带。接下来介绍程序结构与优化设计，讲解主要的程序结构、程序加速与调试方面的知识。最后介绍不同类

型文件的处理方法，其中还包括文件批量处理的方法。

第 2 部分　科学计算　包括第 7～15 章。本部分介绍一些 MATLAB 求解高等数学知识方面的函数功能。首先介绍线性方程组与超越方程的求解方法，解方程在很多问题中都会遇到，通过本部分读者可以了解不同类型方程的解法。然后介绍数据拟合与插值的 MATLAB 实现，它们是数据处理的常用工具。随后介绍最值问题的解法，因为很多问题的最优结果往往是最大值或者最小值，这部分内容可以帮助读者找到最佳答案。接下来介绍随机数的使用，它们是一些经典算法的基础，如蒙特-卡罗算法、随机布朗运动等。然后介绍微分方程组的求解和一些积分表达式的计算，它们是高等数学的基石。最后介绍常用数学变换的 MATLAB 实现，以及一些特殊函数的 MATLAB 计算方法。

第 3 部分　数据可视化仿真　包括第 16～18 章。本部分主要介绍图形的绘制与编辑、用户图形界面设计方面的知识。首先介绍二维图形的绘制，其中包括绘图函数的功能介绍、图形对象的编辑方法、特殊图形的绘制、多图的布局方法、基本图像处理函数介绍、制作动画的方法，以及图形的保存等知识。随后介绍三维图形绘制方面的知识，如基本函数的介绍、彩色图的绘制与编辑、无色网格曲面的绘制、视角与光照效果的设置、图形的注释等知识。通过这些内容，读者可以掌握常见图形的绘制。最后介绍基本的用户图形界面设计，利用这方面的知识读者可以进行人机交互操作，辅助研究动态过程。

第 4 部分　科学问题编程　包括第 19～27 章。本部分介绍不同专业问题的编程实现。首先介绍基本的建模知识，学习它们可以辅助求解专业问题。然后介绍混沌方面部分现象的模拟，如离散与微分方程中的分岔混沌现象的计算、混沌吸引子、Lyapunov 指数等。接下来介绍分形图形的几种绘制方法，如递归、迭代函数系统、L 系统等。另外，元胞自动机方面的基础知识，如奇偶规则、砂堆规则等，晶体生长的模拟，几种光学现象的模拟，如鱼眼效果、全息、干涉现象等，几种机械运动的仿真，如凸轮与连杆的运动等，MATLAB 在经济学中的应用等都在本部分中给予了详细的介绍。在本部分的最后介绍了几种常用算法原理和程序实现。

本书特色

- 本书是 MATLAB 论坛技术版版主多年实战经验的总结。
- 全程实例解说 MATLAB 数值计算和数据可视化功能，提高读者实用经验。
- 本书以 MATLAB R2008a 版本为例介绍其在数值模拟中的应用，覆盖不同专业问题的解法。
- 本书点面兼顾，覆盖更多专业的问题。
- 配套的光盘可以免去读者烦琐输入代码的工作，提高学习效率。
- 较多过程模拟采用动画方式，使其生动形象。

读者对象

本书是有数值计算编程需求但对编程不熟悉的人士，以及利用 MATLAB 求解复杂科学问题研究人员的参考书目。本书既可以作为学校或培训机构及企业利用 MATLAB 进行数值计算的教程，也可作为读者自学 MATLAB 的参考用书。

本书约定

本书的附图和运行结果可能会与读者实例环境中的操作界面或结果略有差别，这可能是由于操作系统平台、MATLAB 版本不同而导致的，在此特别说明，一切以实际环境为准。

分工与致谢

本书由刘正君博士编著，其中南京师范大学的博士生陈玲玲为第 27 章的编写提供了大量素材，姚新军负责前期的策划与后期质量监控，本书由成都易为科技有限责任公司审校。参与工作的同志有：黄中林、张强林、王晓、王斌、万雷、吴艳、王呼佳、夏慧军、张赛桥、陶林、赵会春、余松、李晓宁、赵滕伦、李佳等。本书的出版得到以下基金的资助：国家自然科学基金（10674038，10604042）、国家基础研究项目（2006CB302901）、中国博士后科学基金（20080430913）、哈尔滨工业大学优秀青年教师培养计划（HITQNJS.2008.027）。作者在此特别感谢家人的大力支持。

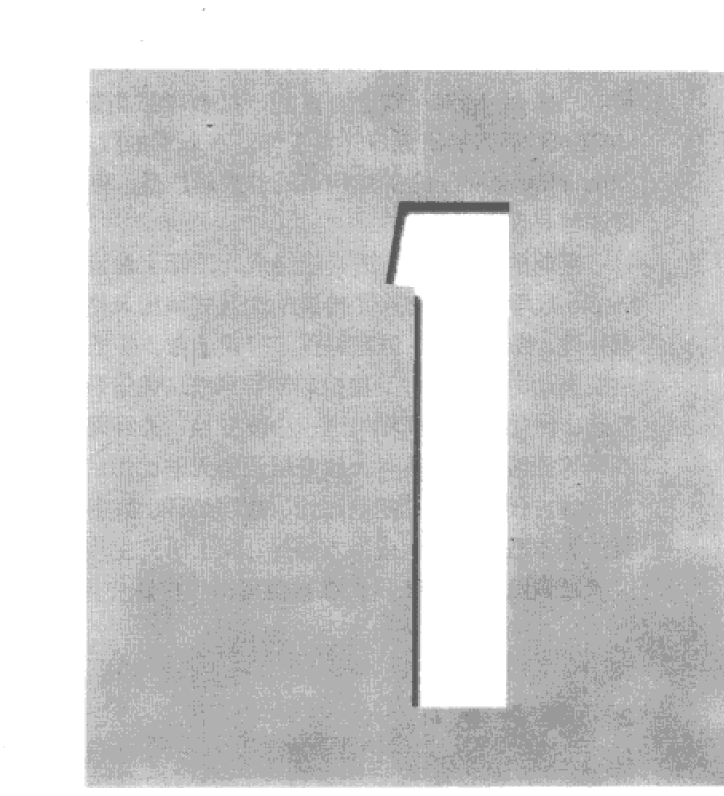
由于时间和水平有限，书中不足之处在所难免，恳请读者批评指正。

目　　录

Part

第1部分 基础篇

- 第1章 MATLAB 科学计算基础
- 第2章 理解 MATLAB 的数据类型
- 第3章 向量与矩阵运算
- 第4章 表达式
- 第5章 程序结构与优化
- 第6章 文件处理



第 1 章 MATLAB 科学计算基础

本章包括

- ◆ **MATLAB 简介** 介绍 MATLAB 的发展和版本 2008a 的功能。
- ◆ **安装说明** 介绍 MATLAB2008a 版本的安装过程。
- ◆ **初学者如何开始使用 MATLAB** 简单介绍初学者应该注意的问题。
- ◆ **如何获取帮助** 介绍寻求帮助的办法。
- ◆ **路径设置与转换** 介绍设置当前路径和搜索路径的方法。
- ◆ **偏好设置** 介绍设置界面风格等方法。
- ◆ **添加工具箱** 总结两种添加工具箱的方法。

本章是本书的第 1 章，将介绍 MATLAB 的基础知识，其中包括 MATLAB 简介、基本知识和函数的使用，以及如何开始使用 MATLAB 等。希望通过本章的介绍，使得读者对 MATLAB 产生兴趣，不断积累使用中的经验和技巧，同时也能和其他人共享你的收获。相信各位朋友能在分享中学习到新东西，作者也尽量把自己掌握的相关知识和体会展现给读者。

1.1 认识 MATLAB

在 1980 年，美国的 Clevel Moler 博士在新墨西哥大学教授线性代数时，发现采用高级语言编程很不方便，因此他开发了 MATLAB（Matrix Laboratory，缩写为 MATLAB）软件，即矩阵实验室。该软件是进行科学计算的软件系统。经过几年试用之后，在 1984 年该软件的公开版本正式推出，它的核心代码是采用 C 语言编写的。在 MATLAB 环境中，矩阵运算编程和代码输入非常容易且简单。后来以 Moler 博士为首的一批数学家与软件专家组建了 MathWorks 软件开发公司，从事 MATLAB 语言的研究与开发工作。在以后的版本中又增添了图形生成功能、图像处理、多媒体和 Simulink 等功能。它的应用范围越来越广泛，随着需求的增加其功能不断扩充，在工程计算的不同领域都表现非常活跃。

目前的 MATLAB 语言已经成为国际上最流行的编程软件，其用户数量多于 Mathematica 和 Maple 数学软件。它除了传统的交互式编程风格外，还提供了丰富可靠的矩阵运算、图形绘制、数据处理、图像处理、语言编程等专用函数，并把函数分类在工具箱之中。MATLAB 广泛地应用于航空航天动力学系统、卫星控制制导系统、通信系统、船舶和汽车、图像信号处理和信号分析、自动控制、优化设计、模糊推理、小波变换、神经网络、时序分析与建模、振动理论、化学统计学、经济学等领域，同时具有一般高级语言无法比拟的优势。在欧美高等院校中，MATLAB 语言已经成为应用线性代数、自动控制理论、数理统计、数字信号处理、时间序列分析、动态系统仿真等高级课程的基本教学工具，成为本科生、硕士研究生、博士研究生必须掌握的基本技能，同时也是广大研究者所青睐的计算工具，在数值模拟中广泛使用。MATLAB 的主要功能包括：

- 此高级语言可用于技术计算。
- 此开发环境可对代码、文件和数据进行管理。
- 交互式工具可以按迭代的方式探查、设计及求解问题。
- 数学函数可用于线性代数、统计、傅里叶分析、筛选、优化以及数值积分等。
- 二维和三维图形函数可用于可视化数据。
- 各种工具可用于构建自定义的图形用户界面。

各种函数可将基于 MATLAB 的算法与外部应用程序和语言（如 C，C++，Fortran，Java，COM 和 Microsoft Excel）集成。

MATLAB 从最初的版本经历多次升级，比如从 5.3 到 6.5。从 2006 年开始，MathWorks 公司将每年进行两次产品发布，时间分别在每年的 3 月和 9 月。版本名用“年份+a”或者“年份+b”来命名，比如 2006a 和 2007b，其中 a 对应 3 月份发布的版本，b 对应 9 月份发布的版本。同时每一次发布都会包含所有的产品模块和一些新增产品。下面介绍一下 2008a 版本的一些信息。

MATLAB R2008a 于 2008 年 3 月 1 日发布，R2008a 版本包括了 MATLAB 和 Simulink 的新特性、两个新产品，以及对 82 种其他产品的更新和 Bug 修复。从 R2008a 开始，MATLAB 和 Simulink 的产品系列需要激活。R2008a 还引入了一个可以提供通用许可管理需求的在线工具。MATLAB 产品系列的新功能包括：

- MATLAB 支持先进的面向对象编程，包括对类和对象、继承、方法、属性、事件和封装的全面支持。
- Optimization Toolbox 支持大型优化问题的内点解算器和并行计算支持。
- Financial Toolbox 用于均值方差组合优化的线性互补程序（LCP）。
- Parallel Computing Toolbox 全面支持 PBS Pro® 和 TORQUE 调度器。
- Statistics Toolbox 的交叉验证、特征选择、准随机号码和最小二乘法。

Simulink 产品系列的新功能包括：

- Simulink 重新设计的多平台 Library Browser（库浏览器）。
- Real-Time Workshop Embedded Coder 支持符合 AUTOSAR 的代码生成。
- 在 M-Lint code analyzer 和 Simulink® Design Verifier 上 EmbeddedMATLAB™ 语言子集功能的代码检测功能。
- IEC 61508 类指南可检查 Simulink Verification and Validation 上的关键安全系统。
- Simulink Fixed Point 中支持可自动转换浮点模型的 Fixed-Point Advisor。
- Communications Blockset 对调制器、解调器、编码器和解码器功能的定点支持。

MATLAB 是一种用于算法开发、数据可视化、数据分析以及数值计算的高级技术计算语言和交互式环境。使用 MATLAB，可以较使用传统编程语言（如 C，C++和 Fortran）更快地解决技术计算问题。MATLAB 的应用范围非常广，包括信号和图像处理、通信、控制系统设计、测试和测量、财务建模和分析以及计算生物学等众多应用领域。附加的工具箱（单独提供的专用 MATLAB 函数集）扩展了 MATLAB 环境，以解决这些应用领域内特定类型的问题。MATLAB 提供了很多用于记录和分享工作成果的功能，可以将用户的 MATLAB 代码与其他语言和应用程序集成，开发用户的 MATLAB 算法和应用。

1.2 安装说明

MATLAB 的安装步骤如下：

step 1 把 MATLAB 软件的光盘放置在光驱中，找到 setup.exe 文件双击运行后会出现如图 1.1 所示的界面，单击 Next 按钮将会出现如图 1.2 所示的界面。

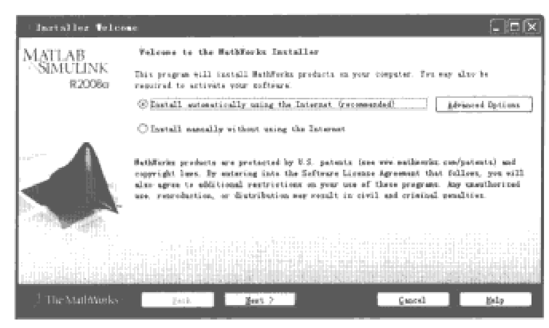


图 1.1 安装初始界面

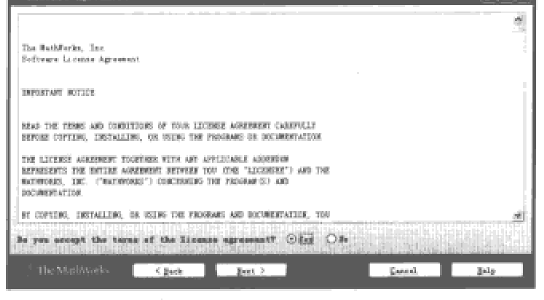


图 1.2 版权协议界面

step 2 选择 Yes 单选项再单击 Next 按钮出现如图 1.3 所示的界面。在输入正确的注册码后单击变为可用状态的 Next 按钮，出现如图 1.4 所示的界面。

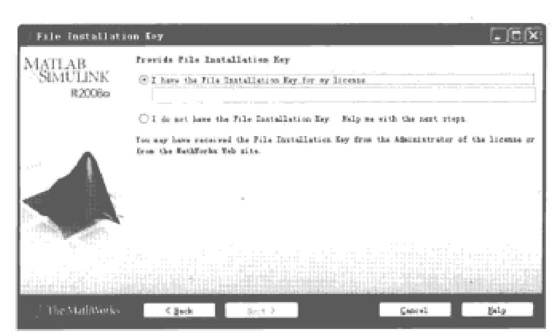


图 1.3 输入注册码界面

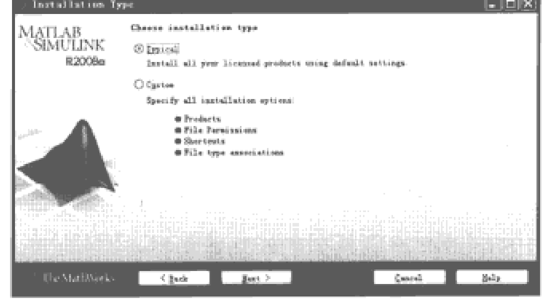


图 1.4 安装方式的选择

step 3 这里有两种安装方式：Typical 和 Custom。如果用户不熟悉 MATLAB 中工具箱所具有的功能，建议选择 Typical 方式安装；对于熟悉 MATLAB 工具箱的用户建议选择 Custom 方式，这种方式下用户可以选择自己需要的工具箱进行安装。单击 Next 按钮出现如图 1.5 所示的界面。

step 4 在如图 1.5 所示的界面中，用户可以根据自己的软件安装习惯来设定路径。设置好路径后单击 Next 按钮将会弹出如图 1.6 所示的界面。在该界面中，用户可以使用鼠标把选项前面的钩号去掉，即不安装对应的内容。只选择需要的内容，可以减小安装的空间要求，同时也可以减小 MATLAB 启动时的加载量。用户选择好自己需要的安装内容后单击 Next 按钮继续安装进程，弹出如图 1.7 所示的界面。



图 1.5　设置安装路径的界面

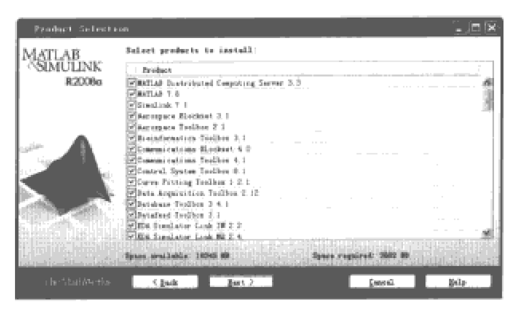


图 1.6　选择模块和工具箱的界面

step 5 用户单击 Next 按钮会弹出如图 1.8 所示的确认安装内容的界面。用户可以检查一下是否有误，如果发现问题可单击 Back 按钮返回前一步重新设置，确认无误后单击 Install 按钮进入复制文件环节和安装进程的界面。在复制完文件后，MATLAB 的安装程序会提示用户安装完成，至此 MATLAB 就成功安装了。

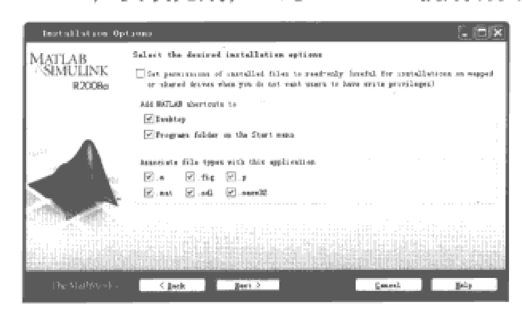


图 1.7　快捷方式和支持文件类型



图 1.8　确认安装内容的界面



有的版本在安装的时候可能需要 Java 虚拟机，相应文件可以在如下网址下载：ftp://ftp.mathworks.com/pub/tech-support/solutions/s26356/msjavx86.exe。

1.3　初学者如何开始使用 MATLAB

启动 MATLAB，就可以进入 MATLAB 的默认界面，如图 1.9 所示。在默认界面中由当前路径（Current Directory）、命令历史记录（Command History）和命令窗（Command Window）等 3 个窗口组成。用户可以根据自己的习惯在菜单 Desktop/Desktop Layout 下选择需要的窗口风格。

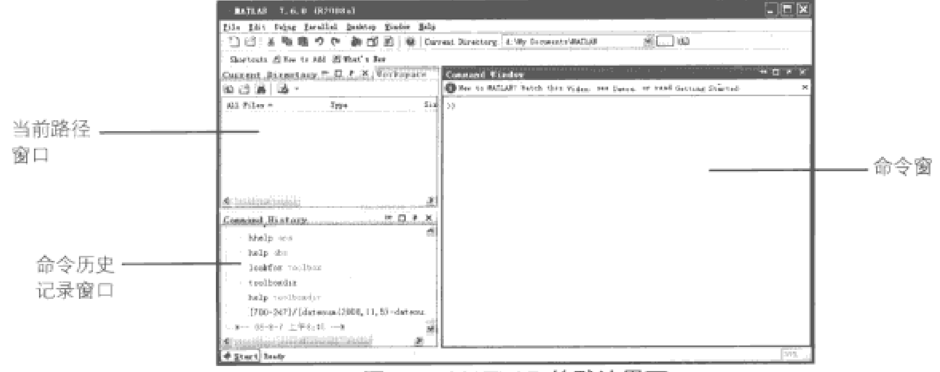


图 1.9　MATLAB 的默认界面

用户可以在 MATLAB 界面的左上角单击白色按钮（标注“新 M 文件”的按钮）打开一个空白的 M 文件编辑窗口，开始输入自己的程序内容。类似文件夹的按钮可以打开当前文件夹来选择要打开的文件。在 Current Directory(当前路径)文本框中可以输入或选择一个路径。初次使用 MATLAB 的用户应该先熟悉整个界面环境，了解各个部分的作用。如果熟悉以前版本，用户只要先熟悉自己常用的功能即可开始工作了（MATLAB 新的版本会增加一些功能，但是以前的一些常用功能还会保留）。

初学者可以通过很短的时间来熟悉上面介绍的 MATLAB 基本操作方面的知识，然后就可以开始编写程序来解决自己的问题。一些高级的功能可以日后再去了解，而且这些功能随着版本的不同会略有差异。

1.4 如何获取帮助

用户可以有 3 种方式获取 MATLAB 中的使用帮助，即本地帮助、求助朋友和网络帮助。求助的原则是先自己想办法，实在想不出来可以考虑使用求助方式找办法。

1.4.1 本地帮助

我们可以从 MATLAB 中获得帮助，对于 MATLAB 中函数（有的书叫做命令，本书统一称为函数）帮助信息可以用下面的方式调用：

```
help functionname
```

比如用户在命令窗中输入 help dir 将会弹出如下信息，如图 1.10 所示。

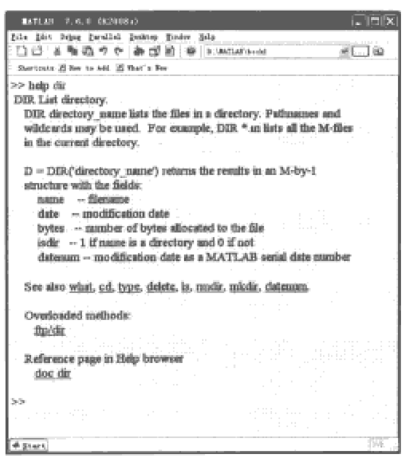


图 1.10 获取帮助信息

这个帮助信息中含有函数功能的描述、调用格式、参数意义、示例、相关函数、help 窗口的链接等信息。用户可以通过英文的含义来了解函数的使用。对于有下画线标识的相关函数，用户可以用鼠标单击得到相应函数的帮助信息。例如，单击“doc dir”将会出现如图 1.11 所示的帮助窗口。

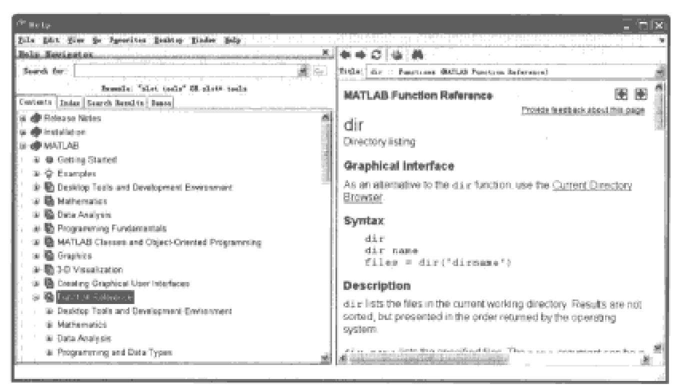


图 1.11 帮助窗口

用户可以在帮助窗口中看到一些非文本的内容，它如同一本电子书，尽可能地详细介绍 MATLAB 的使用。这是一个比较好的帮助手段，用户可以选择合适的英文关键词来搜索需要的信息。对于一些基础的问题，用户完全可以在这里找到答案。

1.4.2 网上求助

通过网上求助可以帮助用户解决一些难题。用户可以借助 baidu 和 google 搜索引擎来搜索，网址分别为 http://www.baidu.com 和 http://www.google.com。

搜索之前用户应该提炼出问题中的关键词，在搜索引擎中输入这些关键词，然后在搜索结果中寻找适合自己问题的答案。目前网络资源极大丰富，一般问题的答案或者很多源代码都可以免费下载到。

此外一些高校的 BBS 和 MATLAB 技术论坛还提供了关于 MATLAB 使用的交流版面，用户可以注册并登录参与交流。

1.5 路径设定与转换

用户可以使用如下方法改变启动后的当前路径：在命令窗中输入 edit MATLABrc 后会打开 matlabrc.m 文件，用户把“cd newpath”（其中 newpath 是用户希望的完整路径名）添加到该文件的最后一行，保存后将 matlabrc.m 文件关闭即可。在下次启动 MATLAB 的时候，默认当前路径就是 newpath 了。程序文件的管理原则是分类保存，尽量将相关的文件保存在一起，以便于日后查找。在运行程序的过程中，用户可以使用函数 cd 在不同路径之间转换。

MATLAB 对于输入内容（如 fname）的搜索是按下面的顺序进行的：先把 fname 作为一个变量来搜索，如果在 workspace 和内存中找不到 fname，再把 fname 作为 MATLAB 内建函数进行搜索，如果还是找不到就在当前路径中搜索 fname.m 和 fname.mex 文件。如果上述范围均未找到，MATLAB 会提示用户，fname 是未定义的函数或者变量。

在 MATLAB 安装的根目录中，部分路径是 MATLAB 的默认搜索路径，它们在 pathdef.m 文件中定义。只有在搜索路径内存储的函数才能被用户调用。这里先介绍添加一个路径到 MATLAB 的搜索路径的好处。用户只需要把若干程序文件存储在一个文件夹中，就可以在其他路径调用这些函数。举例来说，假设用户在 D 盘上新建了两个文件夹 DD1 和 DD2，这两个文件夹的程序都需要调

用同一类函数（它们存储在同一个文件夹中，简称工具箱）。此时，如果用户未把该工具箱添加到 MATLAB 的搜索路径下，则需要分别把工具箱中所有用到的文件都复制到目录 DD1 和 DD2 下，主程序才能正常运行。这显然浪费空间和用户的时间，所以 MATLAB 提供了一个搜索路径功能（默认在 MATLAB 安装目录下的 toolbox 中）。用户只要把工具箱对应的整个文件夹复制到搜索路径对应的目录中，同时把该路径正确添加到搜索路径中，就可以在 DD1 和 DD2 中使用这个工具箱了。下面就以 MATLAB 安装目录下的 toolbox 目录作为默认的添加路径进行详细说明。

在 MATLAB 中进行搜索路径设置的函数有 path，addpath 和 rmpath。下面将详细介绍这几个函数。

函数 path 可以设置搜索路径，其调用格式为：

```
path                              格式 1
p = path                          格式 2
path(path, 'newpath')             格式 3
```

p 是所有搜索路径构成的字符串，不同路径之间用分号隔开。newpath 是新路径的名称。格式 1 将把所有的搜索路径显示到命令窗中。格式 3 可以把一个新的路径添加到搜索路径中，位置是所搜路径的最后面，当 MATLAB 退出后这个添加的路径将被删除，而默认的搜索路径保存在 pathdef.m 文件中。如果用户想把某个路径加入默认的搜索路径，可以把该路径加入 pathdef.m 文件中，注意要以字符串形式输入。

函数 addpath 可以把路径加到搜索路径的最前面，其调用格式为：

```
addpath dirname                   格式 1
addpath dir1 dir2 dir3 ...        格式 2
```

dirname 和 dir1，dir2，dir3 等是路径名。格式 1 把 dirname 路径加到搜索路径的最前面。格式 2 可以把多个路径加到搜索路径的最前面。这里输入的路径必须是已经存在的路径。下面是相应的例子：

```
addpath D:\works
addpath C:\ D:\
```

函数 rmpath 可以用来从搜索路径中删除一个或者多个路径，其调用格式为：

```
rmpath dirname              格式 1
rmpath dir1 dir2 dir3       格式 2
```

dirname 和 dir1，dir2，dir3 等表示搜索路径中的路径名。格式 1 和格式 2 可以分别用来删除一个或者多个搜索路径。这里要删除的路径名要求是搜索路径中的路径，如果该路径不存在于搜索路径中，将提示：

```
>> rmpath D:\works\qq
Warning: "D:\works\kk;" not found in path.
```

下面是使用 rmpath 的例子。

```
rmpath D:\works\kk
rmpath C:\abc D:\ab
```

用户还可以使用 pathtool 来实现搜索路径的设置，直接在命令窗中输入 pathtool 就可以弹出

相应的操作窗口，如图 1.12 所示。

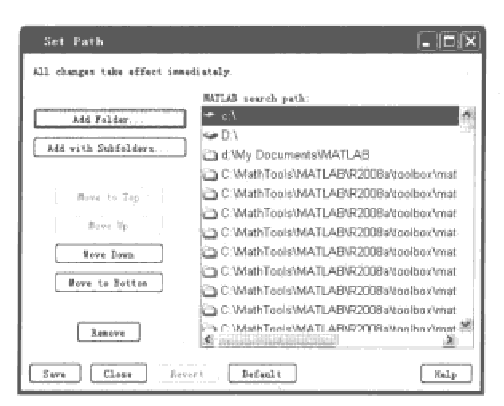


图 1.12　设置路径操作界面

在这个界面中，完全实现了前面介绍的 path，addpath 和 rmpath 的功能，同时用户还可以调整搜索路径的顺序。

1.6　偏好设置

用户可以在如图 1.13 所示的设置窗口（选择菜单 File/Preferences 即可弹出该窗口）中设置命令窗和 Editor 窗口的字体、字号、背景颜色等信息。这里需要提示一下用户，一些字体会使得中文不能正常显示，只要避开这些字体就可以了。

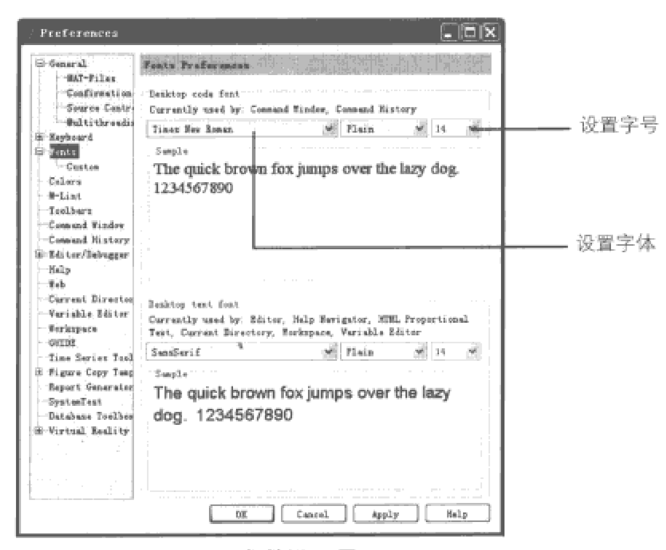


图 1.13　参数设置界面

在图 1.14 中，建议用户将 Editor/Debugger 项下的自动保存（Autosave）选项关闭（即不选择 Enable autosave in the MATLAB Editor 复选框），否则将在当前路径下自动生成一个与 M 文件内容一样的 asv 文件。一般情况 M 文件是不会被损坏的，除非用户不经意删除。

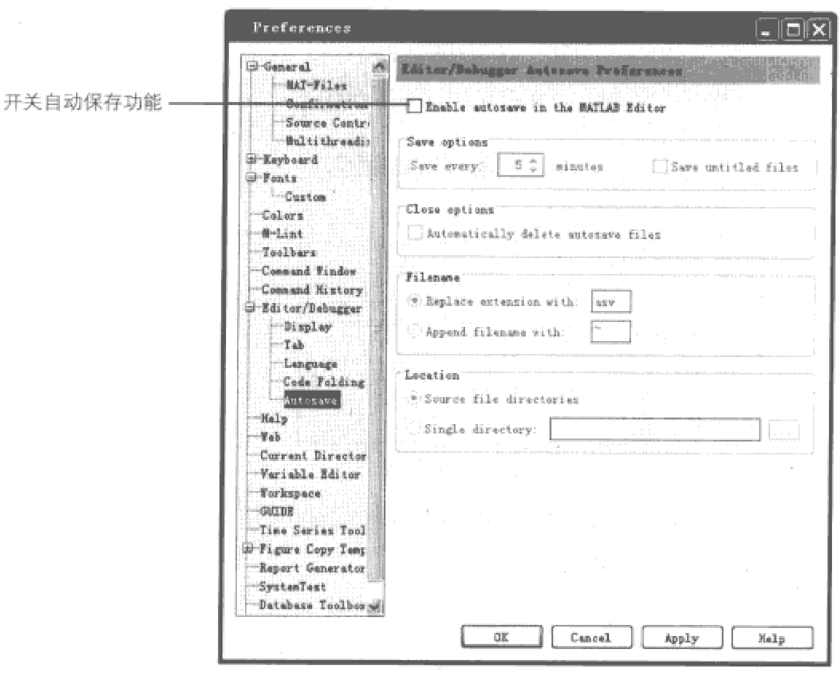


图 1.14　关闭自动保存功能

除了自动保存之外，用户还可以使用函数 pcode 来加密程序内容，其调用格式为：

```
pcode filename
```

filename 是文件名。执行完 pcode 语句后可以得到一个扩展名为 p 的文件，这个 p 码文件具有原来文件的功能（使用者可以运行、调用 p 码文件），但是即使用户自己也无法查看其中的内容。这样的文件可用于一些需要隐藏程序内容的场合，比如用户把自己的文件给别人使用，但不希望别人知道其中的内容。

对于当前路径也可以设置相应的参数，如最多的当前路径数目、显示的文件类型、自动刷新时间等信息。对于 workspace 的设置有：最大显示元素数目以及对于 NaN 值的处理方式。对于 GUIDE 的设置有工具条显示、组件名称、窗口文件扩展名、窗口文件路径名和回调函数等选项。

用户可以使用函数 quit 和 exit 来退出 MATLAB 环境，它们可以用于程序结束的位置，这样程序计算完毕后 MATLAB 就可以自行退出了。利用下面的语句可以实现自动关机：

```
dos('shutdown -s -t 6')
```

上述语句的功能是 6 秒后关机。这里利用函数 dos 调用操作系统中的 shutdown.exe。在 Windows XP 系统中有 shutdown.exe 文件，而在 Windows 2000 系统中没有这个应用程序，用户如果需要可以从 XP 系统中复制过去使用。这个功能可以用于长时间执行的程序中，如果用户离开机器去做其他事情，相应程序的计算结果也自动保存下来，以便用户开机后查看。

在 2008a 版本中还可以把 Editor 窗口和命令窗放在一起，用户可以按图 1.15 所示操作。

用户可以用鼠标单击右上角的 ↘ 按钮把 Editor 窗口和命令窗放在一起，如图 1.16 所示。

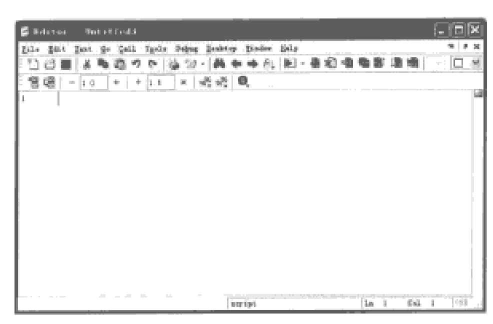


图 1.15 M 文件编译窗口

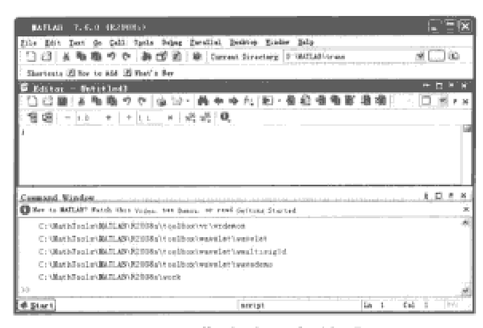


图 1.16 集合在一起的界面

在图 1.16 所示的窗口中，上面程序运行的内容如果出现错误，可以在下面的命令窗中显示出来。这个风格类似于 VC++语言，这样的风格不会因 MATLAB 重新启动而改变。如果用户单击 Editor 窗口右上角的 ↗ 按钮，Editor 窗口将和命令窗分开。用户可以根据自己的喜好来设置。在 Editor 窗口右上角有 5 个按钮，用来控制多个 M 文件的布局，如图 1.17 所示。

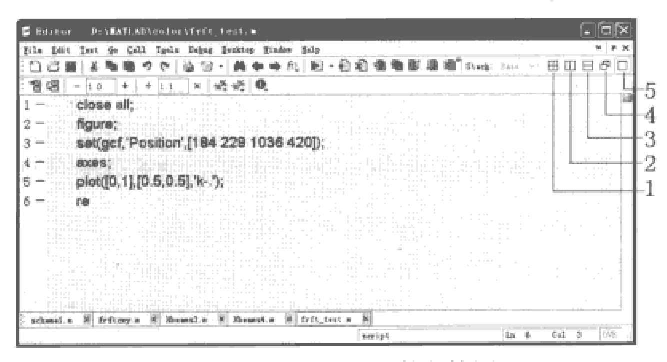


图 1.17 Editor 窗口控制按钮

其中第 5 个按钮是默认风格，第 1，2，3，4 个按钮对应的界面风格如图 1.18 所示。

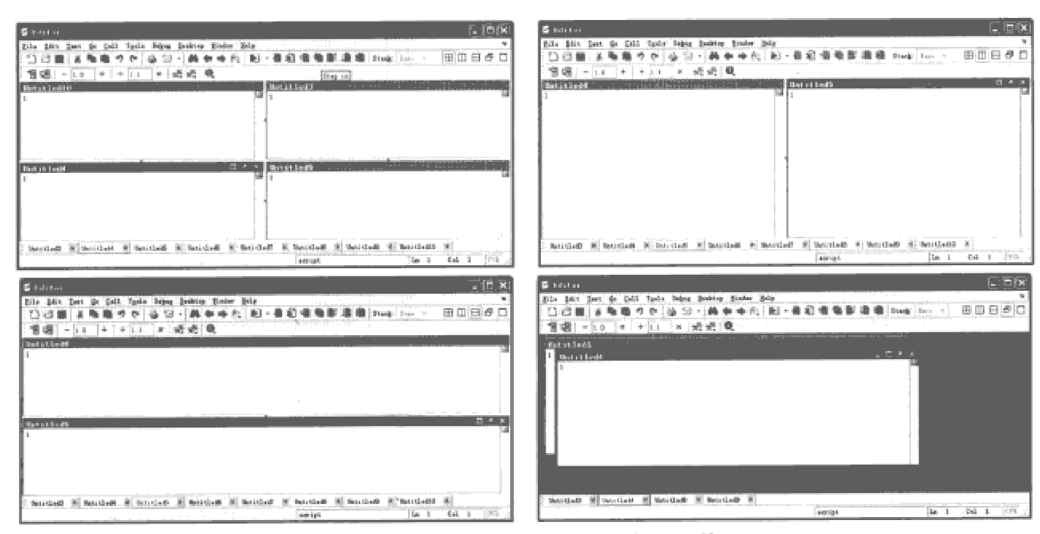


图 1.18 多个 Editor 窗口的布局风格

图 1.18 所示的界面风格在比较不同文件的结果时非常方便。可以用鼠标单击带有绿色三角形

的按钮或者按快捷键 F5 执行 M 文件。

1.7 添加工具箱

为了扩充 MATLAB 的功能，用户可以自己添加新的工具箱。下面介绍两种方法：

- 如果是安装光盘上的工具箱，重新执行安装程序，选中相应的工具箱即可。
- 需要把新的工具箱复制到 MATLAB 安装根目录中的 toolbox 文件夹下，然后用 addpath 或者 pathtool 把该工具箱的路径添加到 MATLAB 的搜索路径中（用户还可以在 pathdef.m 文件中把该路径设置为默认路径），最后用 which newtoolbox_function 来检验新工具箱中的文件是否可以访问。如果能够显示新设置的路径，则表明该工具箱可以使用了。用户也可以查看工具箱中的 readme 文件。

如果用户自己编写了一些文件且经常需要调用它们，可以考虑把它们作为一个工具箱。用户把它们存放于一个文件夹中，然后作为一个工具箱保存在 toolbox 文件夹下，并且按上面介绍的办法进行搜索路径设置。

下面以 MathWorks 提供的工具箱 Submodular Function Optimization 为例说明工具箱安装的过程。该工具箱可以从网页 http://www.mathworks.com/matlabcentral/fileexchange/20504 下载得到。

step 1 下载压缩文件 SFO.zip，解压后复制到 MATLAB 安装根目录中的 toolbox 文件夹下。

step 2 在命令窗中输入 pathtool 后可以弹出如图 1.19 所示的对话框，单击 Add Folder 按钮，弹出如图 1.20 所示的对话框，利用鼠标选择路径 C:\MathTools\MATLABR2008a\toolbox\SFO。

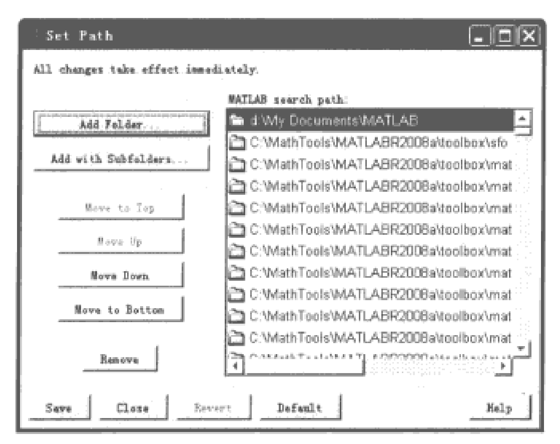


图 1.19 路径设置



图 1.20 路径选择

选中路径之后单击“确定”按钮，再单击如图 1.19 所示对话框中的 Save 按钮，关闭对话框。

step 3 在命令窗中输入下述语句检查工具箱是否安装成功。

```
>> which sfo_logdet
```

MATLAB 输出下述内容就表示工具箱安装成功了。

```
C:\MathTools\MATLABR2008a\toolbox\sfo\sfo_logdet.m
```



上面的路径中“C:\MathTools”是 MATLAB 安装的根目录，如果读者选择其他路径，这里应相应改变。

1.8 小结

本章主要介绍了 MATLAB 的基本知识，介绍了初次接触 MATLAB 时应该了解的一些问题。首先介绍了 MATLAB 的发展以及 2008a 版本的功能；然后以版本 2008a 为例介绍了 MATLAB 的安装过程，围绕 MATLAB 界面介绍了相关组件的功能，使用户熟悉该软件的操作环境；总结了用户获取帮助的 3 种方式，即本地帮助、求助朋友和网络帮助，促进用户在编程中尽快解决所遇到的问题；接下来介绍了设置当前路径和搜索路径的方法，并在偏好设置部分中介绍了一些设置界面的方法，使界面成为用户自己喜欢的风格；最后介绍了添加新工具箱的两种办法。

◆　◆　◆

第 2 章　理解 MATLAB 的数据类型

本章包括

- ◆ **double 型数据**　介绍 double 型数据及其相关函数的使用。
- ◆ **字符串**　介绍字符串定义和相关字符串处理函数。
- ◆ **cell 结构**　介绍 cell 数据的建立和相关的处理函数。
- ◆ **结构型**　介绍结构型数据及相关的函数。
- ◆ **8 位整型数据**　介绍 8 位整型数据相关知识。
- ◆ **不同数据类型之间的转化**　概括了不同数据类型的转化函数。
- ◆ **变量与常量**　介绍了全局变量及 MATLAB 中的常量。

本章将介绍 MATLAB 中的数据类型及不同数据类型之间的转化方法。熟悉这些数据类型对于灵活编写程序具有重要的辅助作用。与其他高级编程语言相比，MATLAB 的数据类型定义起来要简单得多。在 MATLAB 中，常用的数据类型有双精度型（double）、字符型（char）、细胞型（cell）、结构型（struct）、8 位整型（uint8）等 5 种类型，其中在编程中最常用的数据类型是双精度型和字符型数据。下面将详细介绍这些数据类型。

2.1　double 型数据

双精度数据类型是数据计算中最常用的数据类型，在 MATLAB 中可以简单地定义，如：

```
N = 2;
b = 1.5;
c = 2+3i;
c1 = 3+2*i;
```

用户可以根据需要来定义实数和复数，其中 i 和 j 在 MATLAB 中用来表示虚数单位 $\sqrt{-1}$。如果用户在程序中把 i 或者 j 定义为一个数值，如：

```
i = 2;
```

则以后用户用下面两种方式定义一个复数是有区别的。

```
C1 = 1+2i
C2 = 1+2*i
```

输出如下：

```
C1 =   1.0000 + 2.0000i
C2 =     5
```

其中 C1 和 C2，一个为复数，一个为实数，二者差异很大。为了避免这个问题出现，用户可

以不对 i 和 j 重新赋值。通过如表 2.1 所示的 format 函数可以控制 double 型数据的显示格式。

表 2.1 format 函数控制 double 型数据的显示格式

语句	功能
format short	显示带有 4 位小数的定点数，而且 MATLAB 重新启动后置为该格式，如 3.1416
format long	显示带有 14 位小数的定点数，如 3.141592653589793
format short e	显示带有 4 位小数的浮点数，如 3.1416e+000
format long e	显示带有 14 位小数的浮点数，如 3.141592653589793e+000
format short g	显示带有 4 位小数的浮点数或者定点数，如 3.1416
format long g	显示带有 14 位小数的浮点数或者定点数，如 3.14159265358979
format hex	显示为 16 进制数，如 3ff0000000000000
format bank	显示为货币格式的数，即带有 2 位小数，如 3.14
format rat	显示为有理数，如 355/113
format +	正数为+，负数为-，0 为空格，相当于符号函数，复数是按实部计算
format compact	显示为压缩格式，类似于 format +方式
format loose	显示为稀疏格式，类似于 format +方式

函数 isfloat 可以用来判断变量的数据类型是否为浮点型数据，其调用格式为：

```
isfloat(A)
```

参数说明： A 是输出的变量。当 A 为浮点型数据时，返回值 1，否则为 0。

如果用户需要显示特定位数的小数，可以使用 vpa 函数。该函数用法将在 4.4.1 节中详细介绍。MATLAB 函数提供了 4 种取整函数，即 ceil （向正无穷取整）、fix （向零取整）、floor （向负无穷取整）和 round（四舍五入），它们直接可以作用于 double 型数据，这里不再赘述。不清楚的地方，用户可以参阅帮助信息。

2.2 字符串

字符串可以用来表示一些 MATLAB 函数的属性值，并用于显示中英文内容等。本节来介绍字符串的定义以及处理字符串的相关 MATLAB 函数。

2.2.1 字符串的定义

常用的定义字符串的方法是用两个单引号把字符串内容“夹”起来，比如用户可以用下面的方式来定义字符串。

```
s1 = 'Harbin, 黑龙江'
s2 = char(33:128)
```

输出如下：

```
s1 =Harbin, 黑龙江
s2 =
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmn
opqrstuvwxyz{|}~ ?
```

其中，s2 包含大小写英文字母、空格和一些常见的符号，有的时候可以用这种方式来得到难以输入的字符。比如：

```
s1 = ['b=a',char(39)]
s2 = ['b=a''']
```

上面的语句可以得到：

```
s1 =
b=a'
s2 =
b=a'
```

在 MATLAB 中要使单引号出现在字符串中，必须用 2 个连续单引号才能得到这样的结果。或者用户也可以考虑用 char(39)的方式来得到，这种方式对于初次使用 MATLAB 的用户更容易接受和理解。

2.2.2 字符串操作函数

前一小节介绍了字符串定义，本小节来介绍 MATLAB 提供的常用字符串操作函数，它们在程序设计中有重要作用。

（1）函数 eval 可以用来执行用字符串表示的表达式，该函数的调用格式为：

```
eval(S);
[X,Y,Z,...] = eval(S);
```

参数说明： S 是字符串的名称，也可以是一个完整的字符串表达式。X，Y，Z 等代表执行 eval 函数之后相应的输出内容。

例 2-1： 调用函数 eval 来操作字符。

```
a = 1;
b = 2;
s1 = 'eig(magic(3))';  % 表达式的含义是计算 3 阶魔方矩阵的本征向量和本征值
eval('c1=a+b')
[V,D]=eval(s1)
```

输出内容如下：

```
c1 =     3
V = -0.5774   -0.8131   -0.3416
    -0.5774    0.4714   -0.4714
    -0.5774    0.3416    0.8131
D =   15.0000         0         0
       0    4.8990         0
         0         0   -4.8990
```

可见 eval 函数执行了字符串 s1 的内容后把输出结果赋给了 V 和 D，相应的语句与下面的语句是等价的。

```
eval('[V,D]=eig(magic(3))')
```

同时为了加速程序的执行速度，用户可以增加分号使中间的结果不显示。此外，eval 函数可以在批处理结构中有很好的应用，后面的章节会介绍这方面的应用例子。

（2）函数 deblank 可以去掉字符串末尾的所有空格，该函数的调用格式为：

```
R = deblank(S);
```

参数说明： R 是处理后的字符串。S 是输出的字符串。

例 2-2： 调用 deblank 函数的例子。

```
length(deblank('MATLAB   '))
```

输出为：

```
ans =     6
```

通过上面的输出结果我们可以发现 MATLAB 后面的空格被删除了。

（3）函数 findstr 可以用来在长字符串中查找一个短的字符串，并返回相应的位置，其调用格式如下：

```
P = findstr(S1,S2);
```

参数说明： P 是返回的所查找字符首字母的位置。S1 表示长字符串，S2 是目标字符串。

例 2-3： 调用 findstr 函数的例子。

```
S1 = 'How much wood would a woodchuck chuck?';
P1 = findstr(S1,'mu')
P2 = findstr(S1,'wo')
```

输出为：

```
P1 =     5
P2 =    10    15    23
```

（4）函数 isstr 可以用来判断变量是否为字符串，其调用格式为：

```
isstr(S);
```

参数说明： S 是输入的检测变量。当 S 是一个字符串的时候，返回值为 1，否则为 0。

例 2-4： 下面是调用 isstr 函数的一个例子。

```
S = 'sda';
v=isstr(S)
```

输出为：

```
v =     1
```

（5）函数 isletter 可用来判断字符串中的各个元素是否为字母，其调用格式如下：

```
v = isletter(S);
```

参数说明： v 是一个由 0 和 1 组成的向量，当字符串中元素是一个字母的时候，对应位置的返回值为 1，否则为 0。S 是一个待检测的字符串。

例 2-5： 调用 isletter 函数判断是否为字母。

```
S = 'H-I-T and Harbin';
```

```
v = isletter(S)
```

输出为:

```
v = 1    0    1    0    1    0    1    1    1    0    1    1    1    1    1
1
```

（6）函数 isspace 可以用来判读字符串元素是否为空格符，其调用格式为:

```
v = isspace(S);
```

参数说明： v 是返回的变量，它是一个由 1 和 0 组成的向量。当字符串中的元素是一个空格符时，对应位置的返回值是 1，否则为 0。S 是输入的字符串。

例 2-6： 调用 isspace 函数来判断是否为空格符。

```
S = 'A and B';
v = isspace(S)
```

输出为:

```
v =    0    1    0    0    0    1    0
```

（7）函数 lower 和 upper 可以把字符串中的字母转为小写格式和大写格式，调用格式为:

```
s1 = lower(S1);
S2 = upper(s2);
```

参数说明： s1 和 S2 是转化后的字符串。S1 和 s2 是输入的字符串。

例 2-7： 调用 lower 和 upper 函数来实现字母大小写转换。

```
S = 'C or D';
s1 = lower(S)
S2 = upper(S)
```

输出为:

```
s1 = c or d
S2 = C OR D
```

（8）函数 strcat 可以把多个字符串在水平方向上依次连接起来，其调用格式为:

```
T = strcat(S1,S2,S3,...);
```

参数说明： T 为连接后的字符串。S1，S2，S3 等是输入的字符串。

例 2-8： 调用 strcat 函数连接字符串。

```
S1 = 'Red';
S2 = 'Green';
S3 = 'Blue';
T = strcat(S1,',          ',S2,',            ',S3)
```

输出为:

```
T = Red,Green,Blue
```

可见在每个字符串后的空格符被过滤掉了。如果用户需要这些空格，可以用下面两种方式来替换上面的语句。

```
T1 = [S1,',          ',S2,',       ',S3]
T2 = strcat(S1,',',char(127*ones(1,9)),S2,',',char(127*ones(1,7)),S3)
```

输出的 T1 和 T2 为：

```
T1 =Red,          Green,       Blue
T2 =Red,            Green,        Blue
```

（9）函数 strvcat 可以把多个字符串按竖直方向连接起来，其调用格式为：

```
S = strvcat(T1,T2,T3,...);
```

参数说明： S 是返回的连接结果。T1，T2，T3 等是输入的字符串。输出结果按左对齐方式排列，右侧如果字符数目不匹配，则以最长长度为准，字符串长度短的行用空格符补齐。用户可以用 strjust 来更改对齐方式，该函数用法稍后介绍。

例 2-9： 调用 strvcat 函数按竖直方向连接字符。

```
T1 = 'beijing1000';
T2 = 'Harbin1';
T3 = 'China1';
S = strvcat(T1,T2,T3)
```

输出为：

```
S =  beijing1000
Harbin1
China1
```



这个例子中，T1，T2 和 T3 是 3 个长度不同的字符串，函数 strvcat 把它们排为 3 行字符矩阵，各行字符串左对齐，右侧使用空格符补齐。

（10）函数 strcmp 可以用来进行字符串的比较，其调用格式为：

```
TF = strcmp(S1,S2);
```

参数说明： TF 是返回值。S1 和 S2 是输入的两个字符串。当两个字符串的内容完全一样时，TF 等于 1，否则 TF 等于 0。

例 2-10： 调用 strcm 函数比较字符串。

```
S1 = 'ab';
S2 = 'Ab';
S3 = 'ab';
TF1 = strcmp(S1,S2)
TF2 = strcmp(S1,S3)
```

输出为：

```
TF1 =     0
TF2 =     1
```



这里字符串 S1 和 S3 相同，而 S1 和 S2 不相同。利用函数 strcmp 来判断字符串是否相同，返回结果为 0 表示不相同，为 1 表示相同。

（11）函数 strcmpi 可以用忽略英文字母大小写的方式来比较字符串，其调用格式为：

```
TF = strcmpi(S1,S2);
```

参数说明： TF 是返回值。S1 和 S2 是输入的两个字符串。当两个字符串的内容（除字母大小写外）一样时，TF 等于 1，否则 TF 等于 0。

例 2-11： 用函数 strcmpi 以特定的方式来比较字符串。

```
S1 = 'ab';
S2 = 'Ab';
S3 = 'ab';
TF1 = strcmpi(S1,S2)
TF2 = strcmpi(S1,S3)
```

输出为：

```
TF1 =     1
TF2 =     1
```



严格地讲，S1 和 S3 是相同的，而 S2 和另外两个字符串不相同，但是在忽略大小写的情况下，3 个字符串就一样了，所以利用函数 strcmpi 返回了相等的结果。

（12）函数 strjust 可以用来调整字符串矩阵对齐的方式，其调用格式为：

```
T = strjust(S,mode);
```

参数说明： T 是重新排列后的结果。S 是字符串矩阵。mode 是一个字符串，用来描述字符串的对齐方式，有 3 个可选值，即 right 表示右对齐，left 表示左对齐，center 表示居中对齐。

例 2-12： strjust 调整字符串矩阵对齐的方式。

```
S = ['abcdefgh';'ss      '];
T1 = strjust(S,'right')
T2 = strjust(S,'left')
T3 = strjust(S,'center')
```

输出为：

```
T1 =                      T2 =             T3 =
Abcdefgh                  abcdefgh           abcdefgh
      Ss                  ss                    ss
```



这里字符串 S 的输入是左对齐方式，利用函数 strjust 可以得到右对齐、左对齐以及中间对齐的 3 种排列方式，输出结果显示了该函数的功能。

（13）函数 strmatch 可以用来寻找和目标字符串匹配的行，其调用格式为：

```
r = strmatch(str, strarray);
r = strmatch(str, strarray, 'exact');
```

参数说明：r 是返回含有目标字符串的行号组成的列向量。str 是用户定义的目标字符串。strarray 表示搜索范围的字符串矩阵。exact 表示目标字符串 str 要和 strarray 行中的内容完全一致。在进行搜索的时候，strmatch 从各行字符串的开始字符串算起，不匹配就进入下一行进行搜索。

例 2-13：寻找和目标字符串匹配的行。

```
r1 = strmatch('max',strvcat('max','minimax','maximum'))
r2 = strmatch('max',strvcat('max','minimax','maximum'),'exact')
```

输出为：

```
r1 =                          r2 =
    1                             1
    3
```



输入的第 2 个字符串 strvcat('max','minimax','maximum')中，第 1 行内容与 max 一致，而第 3 行中含所有 max 但有其他部分。这里利用 strmatch 进行匹配分析，不使用 exact 控制时，匹配结果返回了 1 和 3，表明第 1 和第 3 行匹配。而利用 exact 控制时，只返回第 1 行和 max 严格匹配的结果。

（14）函数 strncmp 可以用来比较字符串前 N 个字符是否相同，其调用格式为：

```
TF = strncmp(S1,S2,N);
```

参数说明：TF 是返回的比较结果。S1 和 S2 是输入的字符串。N 用来指定比较的范围，其值可以大于两个字符串中的最大长度，此时对字符串进行比较相当于 strcmp 函数。此外，函数 strncmp 可以用 strcmp 来替换，如 TF = strcmp(S1(1:N),S2(1:N))，但此时 N 不能大于两个字符串长度的最小值。当 S1 和 S2 的前 N 个字符串相同时，返回值为 1，否则为 0。

例 2-14：字符串的前 N 个字符是否相同。

```
S1 = 'abc12';
S2 = 'abc23';
TF1 = strncmp(S1,S2,3)
TF2 = strncmp(S1,S2,4)
```

输出为：

```
TF1 =     1                     TF2 =     0
```



这里利用函数 strncmp 比较两个不同字符串前 3 个和前 4 个字符是否相同。输入字符串前 3 个字符相同而后面不同，程序的输出结果表明了这个结论。

（15）函数 strrep 可以实现字符串的查找和替代功能，相当于一些软件如 Word 中的查找、替换功能，其调用格式为：

```
S = strrep(S1,S2,S3);
```

参数说明：S 是替换后的结果。S1 是原始字符串。S2 是要查找的字符串。S3 是代替的字符串。

当在 S1 中找不到字符串 S2 时，将不进行操作。

例 2-15： 串的寻找和替代。

```
S1 = 'This is a good example';
S = strrep(S1,'good','great')
```

输出为：

```
S =This is a great example
```



这个例子把输入字符串中的 good 替换为 great，结果表明这个函数很好地完成了查找目标字符串 good 和替换字符串 great 的功能。

（16）函数 strtok 可以找出字符串中第一个空格符前面的字符串，其调用格式为：

```
T = strtok(S);
```

参数说明： T 是返回的字符串。S 是原始字符串。

例 2-16： 下面是该函数的一个例子。

```
S1 = 'This is a great example, which is ok.';
S = strtok(S1)
```

输出为：

```
S =This
```

（17）函数 texlabel 可以把字符串转换为 TeX 软件的格式，其调用格式为：

```
texlabel(f)
```

参数说明： f 是目标字符串。

例 2-17： 将字符串转换为 TeX 软件的格式。

```
texlabel('sin(sqrt(x^2 + y^2))/sqrt(x^2 + y^2)')
```

输出为：

```
ans = {sin}({sqrt}({x}^{2} + {y}^{2}))/{sqrt}({x}^{2} + {y}^{2})
```

（18）下面给出不同进制的转化函数，具体用法和含义如表 2.2 所示。

表 2.2 进制转换函数列表

调用格式	说明	调用格式	说明
X = bin2dec(B);	二进制转换为十进制	B = dec2bin(D);	十进制转换为二进制
X = oct2dec(C);	八进制转换为十进制	C = dec2oct(D);	十进制转换为八进制
X = hex2dec(H);	十六进制转换为十进制	H = dec2hex(D);	十进制转换为十六进制
N = hex2num(N);	十六进制转换为双精度数据		

然而表 2.1 描述的函数 dec2bin，dec2oct 和 dec2hex 只能把整数转为二进制、八进制和十六进制数。如果用户输入一个小数，这些函数将对整数部分进行转化。比如：

```
B = dec2bin(pi)
```

输出为：

```
B = 11
```

这里只是把 pi 的整数部分“3”转化为二进制数。为此作者编写了 bin2dec_df.m 文件和 dec2bin_df.m 文件（均位于光盘的\Ch02 文件夹内），用于在二进制小数和十进制小数之间进行转化，它们是函数文件，用户可以像调用 MATLAB 自带函数一样使用这两个函数。

例 2-18：下面是调用这两个函数的例子。

```
yd = bin2dec_df('11.00100')
yb = dec2bin_df(pi,5)
```

输出为：

```
yd =    3.1250
yb =    11.00100
```

其中 yb 是字符串数据。函数 bin2dec_df 要求输入的参数为字符串。函数 dec2bin_df 的第一个输入参数为 double 型数据，第二个参数用来指定二进制小数的位数。此外，读者可以把 double 型小数乘 10 的整数次幂取其四舍五入的整数，然后再使用函数 dec2bin，dec2oct 和 dec2hex 来计算，最后在前面加“0.”即可。

（19）函数 bitget 可以用来获取二进制的数位，其调用格式为：

```
C = bitget(A,BIT);
```

参数说明：C 是提取的二进制数位。A 是一个无符号整型数据，如 uint8 和 uint16 等。BIT 用于指定二进制的数位位置。

例 2-19：获取二进制的数位。

```
A = 243;
C = bitget(uint16(A),3:6)
```

输出为：

```
C =      0      0      1      1
```

（20）与 bitget 相关的函数是 bitset，它可以用来设定某个二进制数位的值，其调用格式为：

```
C = bitset(A,BIT);          % 格式 1
C = bitset(A,BIT,V);        % 格式 2
```

参数说明：C 是改变后的整型数值。A 是一个无符号整型数据，如 uint8 和 uint16 等。BIT 用于指定二进制的数位位置。在格式 1 中指定数位的数值将被取非运算的结果，即 1 变为 0，0 变为 1。在格式 2 中指定数位的数值将被 V（这里 V 只能是 1 或者 0）替换。

例 2-20：设定某个二进制数位的值。

```
A = 168;
C1 = bitset(uint8(A),2)
C2 = bitset(uint8(A),3,1)
```

输出为：

```
C1 =  170                                     C2 =  172
```



168 对应的二进制数位数值是 00010101，从左至右依次是从低位到高位。对于 C1，二进制数将变为 01010101，它对应的十进制数是 170；对于 C2，二进制数将变为 00110101，它对应十进制数为 172。

（21）函数 bitand，bitor 和 bitxor 可以用来进行“与”、“或”和“异或”数位操作，它们的调用格式为：

```
C = bitand(A,B);
C = bitor(A,B);
C = bitxor(A,B);
```

参数说明：C 是返回的计算结果。A 和 B 是两个无符号整型数据。

例 2-21：下面是“与”、“或”和“异或”数位操作的例子。

```
A = uint8(167);
B = uint8(211);
C1 = bitand(A,B)
C2 = bitor(A,B)
C3 = bitxor(A,B)
```

输出如下：

```
C1 =  131
C2 =  247
C3 =  116
```



167 对应的二进制数为 11100101（从左至右依次是从低位到高位），而 211 对应的二进制数为 11001011。这两个二进制数进行“与”、“或”和“异或”运算的结果分别为 11000001，11101111 和 00101110，它们依次对应于 131，247 和 116。

2.3 cell 结构

在 MATLAB 中，提供了一种 cell 结构。在这种数据结构中可以允许有不同的数据类型，这一点使得该数据类型不同于 double 型数据和字符串类型。在 cell 结构中可以包括字符串、double 型数据以及 cell 结构。cell 结构也可以有行数和列数。下面来介绍 cell 结构的定义和相关函数。

用户可以使用 3 种办法来建立一个 cell 结构：第一个办法是使用大括号“{}”，用户可以像使用方括号那样来定义向量和矩阵；第二种办法是对 cell 结构的元素逐一赋值；第三种办法是用函数 cell 来建立一个空的 cell 结构。

例 2-22：cell 函数的应用。

```
A = {'ABC',11,'Zhang',12,'Liu',{11,'Hello'}}
B = {};
```

```
B{1,1} = 'KK';
B{2,1} = 121;
B{1,2} = 'HIT'
C = cell(2,3)
```

输出如下：

```
A =     'ABC'    [11]    'Zhang'    [12]    'Liu'    {1x2 cell}
B =     'KK'     'HIT'
        [121]       []
C =     []       []      []
        []       []      []
```

cell 结构中元素可以用 A{m,n}，A(m,n)这样的方式来调用，它们的不同之处在于：A{m,n}得到结果的类型为元素自身的类型，A(m,n)的数据类型为 cell 结构。前面介绍了 cell 结构的定义，下面来介绍 MATLAB 提供的 cell 结构相关函数的用法。

2.3.1 图形化表示 cell 的内容

函数 cellplot 可以画出一个图形来示意 cell 中的内容，其调用格式为：

```
cellplot(C)
```

参数说明： C 为一个 cell 结构。

例 2-23： cellplot 画图。

```
D = {'ABC',magic(2),{1:3,'DEF'}};
cellplot(D)
```

执行上述语句后会弹出如图 2.1 所示的界面。图中橙色表示字符串，红色表示 double 型数据。顺序和结构与 cell 结构中的内容一致。

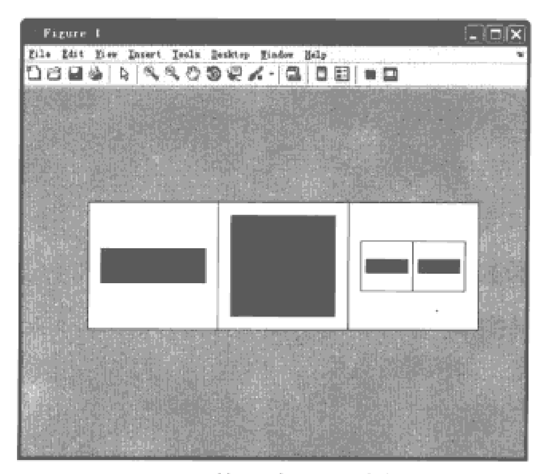


图 2.1　函数 cellplot 得到的图形

2.3.2 检查变量是否为 cell 结构

函数 iscell 可以用来检查变量是否为 cell 结构，其调用格式为：

```
TF = iscell(C);
```

参数说明：TF 是返回的判断结果。C 代表一个变量名。当 C 是 cell 结构时，TF 等于 1，否则 TF 等于 0。

例 2-24：检查变量是否为 cell 结构。

```
Dc = {'ABA',magic(3),'DEF'};
TF1= iscell(Dc)
TF2= iscell(Dc(2))
TF3= iscell(Dc{2})
```

输出为：

```
TF1 =     1                    TF2 =     1                 TF3 =     0
```



本例中定义的 Dc 是一个 cell 结构，提取其第二项需要用 Dc(2)来表示。而利用 Dc{2} 得到的是矩阵 magic(3)的内容，它的数据类型不是 cell，因此函数 iscell 判断的结果为 0。

2.4 结构型

在结构（struct）型数据中，允许用户把不同类型的数据保存到一个整体单元中（MATLAB 中称之为“域”）。对应的不同单元使用一个名字来记录，用户可以根据这个名字来更换和调用其中的数据。下面是建立结构体的例子。

```
St.D = magic(3);
St.s = 'ABC';
St.c = {'aa',1}
```

输出为：

```
St =
    D: [3x3 double]
    s: 'ABC'
    c: {'aa'  [1]}
```

用户还可以用函数 struct 来建立一个结构体，其调用格式为：

```
S = struct('field1',V1,'field2',V2,...);
```

参数说明：S 是返回的结构体。输入参数为“域”的名称（field1，field2，...）以及对应的值（V1，V2，...）。二者需要同时存在，保持对应关系。

例 2-25：结构体的例子。

```
s = struct('type',{'big','little'},'color','red','x',{3 4})
```

输出为：

```
s =
1x2 struct array with fields:
```

```
    type
    color
    x
```

结构体中的内容需要按下面的方式来调用：

```
s.type
```

即以“结构体+逗点+域名”方式来写各个域。在命令窗中输入上述内容后会显示：

```
ans =big
ans =little
```

这里域中数据根据不同类型把每个元素逐一显示出来。我们可以用函数 fieldnames 来得到结构体中的域名称，其调用格式如下：

```
names = fieldnames(S);
```

参数说明： names 是用域名称组成的一个 cell 型数据。S 是结构体的名称。

例 2-26： 获取结构体中的域名称。

```
s = struct('strings',{{'hello','yes'}},'lengths',[5 3]);
names = fieldnames(s)
```

输出为：

```
names =
    'strings'
    'lengths'
```

用户还可以调用 getfield 函数来获取结构体中各个域的内容，其调用格式为：

```
F = getfield(S,'field');
```

参数说明： F 是返回域中的内容。S 是结构体名称。field 用于指定域名。

例 2-27： 获取结构体中各个域的内容。

```
s = struct('strings',{{'Heilongjiang ','Harbin'}},'lengths',[12 6]);
G = getfield(s,'lengths')
```

输出为：

```
G =    12     6
```

结合 fieldnames 函数，用户可以获取不同域的内容。rmfield 函数可以用来从结构体中删除一个域，其调用格式为：

```
S = rmfield(S,'field');
```

参数说明： S 是预处理的结构体的名称。field 用于指定要删除的域名称。

例 2-28： 从结构体中删除一个域。

```
ss = struct('Names',{{'Bill','John'}},'lengths',[4 4]);
ss = rmfield(ss,'lengths')
```

输出为：

```
ss =     Names: {'Bill'  'John'}
```

函数 isstruct 可以用来判断数据是否为结构体，其调用格式为：

```
TF = isstruct(S);
```

参数说明： TF 是返回的判断结果。S 是输入的变量名。

例 2-29： 判断数据是否为结构体。

```
s1 = struct('Names',{{'Zhang','Liu'}},'lengths',[5 3]);
TF1 = isstruct(s1)
TF1 = isstruct(s1.lengths)
```

输出为：

```
TF1 =     1                          TF1 =     0
```

2.5 8 位整型数据

无符号 8 位整型数据的范围是 0~255，数字图像的像素值就对应着这样的数据类型。在 6.5 版本中很多函数不支持 8 位整型数据的运算，使用时需要利用函数 double 把 8 位整型数据转化为双精度数据类型才能计算。而在 2008a 版本中很多函数支持直接对 8 位整型数据进行计算。

8 位整型数据分为两种形式：带符号的 8 位整型数据和无符号的 8 位整型数据，读者可以利用函数 int8 和 uint8 把双精度数据转化为这两种数据格式。带符号的 8 位整型数据的数据范围是 $[-2^7,2^7-1]$。此外还存在着 16 位、32 位和 64 位整型数据，它们的无符号数据类型对应的最小值是 0，最大整数数值依次是：$2^{16}-1$，$2^{32}-1$，和 $2^{64}-1$；而带符号的整型数据对应的范围依次是 $[-2^{15},215-1]$，$[-2^{31},2^{31}-1]$和$[-2^{63},2^{63}-1]$。它们相应的转化函数是 int16，int32，int64，uint16，uint32 和 uint64，其中以 8 位整型数据最为常用。在下一节将介绍整型数据与其他数据类型之间的转化关系。

2.6 不同数据类型之间的转化

在 MATLAB 中查看数据类型可以用 whos 和 class 函数来进行。在命令窗中输入 whos 后可以得到当前 workspace 里面的所有变量名称、大小、字节数、数据类型等信息，这些信息对于用户编写程序和调试程序很重要。特别是还可以利用函数 whos 来查询某个变量的属性。比如：

```
whos s1
```

这里 s1 是变量名，输出下面的信息：

```
 Name      Size            Bytes  Class     Attributes
 s1        1x1               400  struct
```

如果用 whos 查询一个不存在的变量，将不输出任何信息。利用 class 函数可以得到变量的类型，即 whos 查询结果中 class 里面的内容。函数 class 的调用格式为：

```
S = class(OBJ);
```

参数说明： S 是返回的一个由类型组成的字符串。OBJ 是变量名。

下面给出数值数据类型转化的函数及用法说明，具体如表 2.3 所示。

表 2.3 数值数据类型转换函数

调用格式	说明	调用格式	说明
double(X)	转化为双精度型浮点数	single(X)	转化为单精度浮点型数据
int8(X)	转化为带符号 8 位整型数据	uint8(X)	无符号 8 位整型数据
int16(X)	转化为带符号 16 位整型数据	uint16(X)	转化为无符号 16 位整型数据
int32(X)	转化为带符号 32 位整型数据	uint32(X)	转化为无符号 32 位整型数据
int64(X)	转化为带符号 64 位整型数据	uint64(X)	转化为无符号 64 位整型数据

double 型数据和字符串之间可以用 num2str 和 str2num 来转换，如：

```
s = num2str(6.8)
n = str2num('3.6')
```

输出为：

```
s =6.8
n =3.6000
```

此外，字符串变量还可以与 double 型数据进行混合计算，比如：

```
D = 'S'-3.2
```

输出为：

```
D =   79.8000
```

字符串减 0 还可以实现字符串到 double 型数据的转换，如‘S’-0。函数 cellstr，char，cell2struct 和 struct2cell 可以用来在字符串、cell 结构和结构体之间进行转化。它们之间的关系可以用图 2.2 描述。

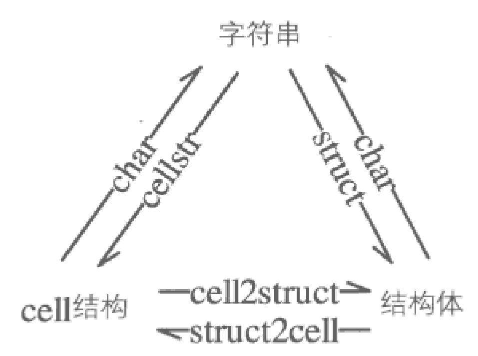


图 2.2 3 种数据类型转化函数关系图

函数 cellstr 可以把字符串转化为 cell 结构，其调用格式如下：

```
C = cellstr(S);
```

参数说明： C 是转化后的 cell 结构。S 是字符串。

例 2-30：将字符串转化为 cell 结构。

```
ss='ssss ccc';
cs1 =cellstr(ss)
```

其中，cs1 的数据类型是 cell。

可以用 cell2struct 函数把 cell 结构转化为结构体，其调用格式如下：

```
S = cell2struct(C,fields,dim);
```

参数说明：S 是转化后的结构体。C 是 cell 结构，fields 用于指定域名称。dim 用于指定维数。

下面是调用该函数的例子。

```
cs2 = cell2struct(cs1,'F1',2)
```

函数 struct2cell 可以把结构体转化为 cell 结构，其调用格式为：

```
C = struct2cell(S);
```

参数说明：C 是转化后的 cell 结构。S 是结构体。

例 2-31：把结构体转化为 cell 结构。

```
S = struct('str',11);
C = struct2cell(S)
```

输出为：

```
C =     [11]
```

2.7 变量与常量

前面介绍了一些数据类型，本节来介绍变量和常量。变量是指其数值在数据处理的过程中可能会发生变化的一些数据量名称，而常量是指在计算过程中数值不发生变化的量。前面介绍的一些数据类型都可以认为是变量，在计算中变量的应用频率高于常量。

如果不声明，MATLAB 中的变量是局部变量。此外用户还可以使用函数 global 来定义全局变量，这种变量可以用于不同函数文件（函数文件的内容将在第 6 章详细介绍）之间的传递。在退出 MATLAB 或者使用函数 clear 清除变量之前，global 定义的全局变量起作用。下面是一个用全局变量计算阶乘的例子。

```
function y = fract;
global N;
y=1;
n=N;
while n>1;
   y=y*n;
   n=n-1;
end
```

把上述程序内容保存为 fract.m 文件，然后在命令窗中输入如下内容：

```
clear
global N;
N=4;
y = fract
```

输出为：

```
y =24
```

需要注意的是，在使用全局变量的整个过程中不能改变全局变量的值，否则容易出错。这种形式的变量已经不提倡使用，用户可以把需要传递的变量作为函数的输入参数，这样就可以避免使用全局变量了。

在 MATLAB 中自带了一些常量，如表 2.4 所示。

表 2.4　MATLAB 自带常量

常量名	意义	常量名	意义
i	虚数单位	j	虚数单位
inf	正无穷大	-inf	负无穷大
pi	圆周率	NaN 或 nan	非数。0/0，inf/inf 等表达式所得结果
eps	浮点数的相对精度，等于 2.2204e-016	realmax	最大正浮点数，等于 1.7977e+308
realmin	最小浮点数，等于 2.2251e-308	intmin	最小整型数据，等于-2147483648

2.8　小结

本章主要介绍了 MATLAB 中常用数据类型的相关知识。首先介绍了 double 型数据的相关知识，包括数据显示格式控制和取整操作等；接下来介绍了字符串数据类型的相关知识，包括字符串的定义、处理方面的一些函数用法；然后介绍了 cell 结构和结构体，主要涉及它们的定义以及相关函数的用法；针对介绍的数据类型，描述了不同数据类型之间的转换函数，熟悉这些内容，用户可以灵活地建立自己的程序；最后介绍了 MATLAB 中的变量和常量，包括全局变量和一些常量。

◆　◆　◆

第 3 章　向量与矩阵运算

本章包括

- ◆ **向量的定义**　介绍几种定义向量的方法。
- ◆ **向量运算函数**　给出与向量运算相关的函数及其用法。
- ◆ **集合的定义及相关运算**　介绍 6 个集合计算函数的用法。
- ◆ **矩阵生成方法**　介绍定义矩阵的方法。
- ◆ **特殊矩阵的生成**　给出一些生成特殊矩阵的函数。
- ◆ **矩阵计算的基本函数**　介绍与矩阵相关的函数及用法。
- ◆ **高维数组**　介绍高维数组及相关的函数。

MATLAB 是一种以矩阵为基本元素的科学计算语句，大多数函数都是支持矩阵变量进行计算的。本章将介绍向量和矩阵的定义与基本运算。熟悉这些知识，对于编写 MATLAB 程序具有重要意义。一定的线性代数知识对于理解本章内容很重要，此外一些高等数学知识也可以帮助掌握和应用本章内容。在 MATLAB 中定义向量和矩阵是比较简单的，不必像一些高级计算机语言那样需要建立一个冗长的循环结构来建立矩阵。

3.1 向量的定义

在 MATLAB 中，用户可以用不同方式来定义向量，本节将详细介绍定义向量的方法。利用冒号可以建立等间隔的向量，常用的格式有以下 3 种：

```
v1 = 1:N;             % 格式 1
v2 = x1:dx:x2;        % 格式 2
v3 = x4:-dx:x3;       % 格式 3
```

参数说明： v1，v2，v3 是返回的向量名。格式 1 中向量的步长等于 1，向量 v1 的最小值和最大值分别是 1 和 N，这种格式常用于循环指标的定义。格式 2 中设定向量间隔的步长是 dx，向量 v2 的最小值和最大值分别是 x1 和 x2。格式 3 中向量的步长等于-dx，它是一个负数，而向量的最大值和最小值分别是 x4 和 x3。当 x1>x2 且 dx>0（或者 x4<x3 且-dx<0）时，返回的向量 v2（或者 v3）是一个空矩阵。所有返回的向量都是行向量。

例 3-1： 生成向量。

```
v1=0:3
v2=1:.1:2
```

输出为：

```
v1 =     0     1     2     3
```

```
v2 =    1.0000    1.1000    1.2000    1.3000    1.4000    1.5000    1.6000    1.7000
1.8000    1.9000    2.0000
```



在 v1 的定义中步长缺省，这时步长等于 1，而向量 v2 的生成语句中，步长等于 0.1。

函数 linspace 可以生成两个数之间的等间隔向量，其调用格式如下：

```
v1 = linspace(x1,x2);       % 格式 1
v2 = linspace(x1,x2,N);     % 格式 2
```

参数说明： v1 和 v2 是返回的向量名。在格式 1 中，x1 和 x2 是向量的两个端点，向量元素个数是 100。在格式 2 中，x1 和 x2 表示向量的两个端点，N 用于指定向量元素个数。当 N 是一个小数的时候，MATLAB 将把 N 进行向 0 取整，即 fix(N)；当 N 是负数的时候，返回的向量 v2 将等于 x1。同时，x1 和 x2 的大小关系任意，MATLAB 自动调整正负步长，x1 作为向量的起点，x2 作为向量的终点。所有返回的向量都是行向量。

例 3–2： 生成向量。

```
v1=linspace(1,2);
v2=linspace(1,2,300);
s1=size(v1)
s2=size(v2)
```

输出为：

```
s1 =     1   100
s2 =     1   300
```



当 linspace 的输入参数 N 缺省时，生成的向量长度是 100，读者在使用的时候还可以特别指定。因为数据较多，这里只是给出了函数 size 计算的向量大小数值。

函数 logspace 可以用来产生一个对数向量，其调用格式如下：

```
v1 = logspace(x1, x2);       % 格式 1
v2 = logspace(x1, x2, n);    % 格式 2
```

参数说明： v1 和 v2 是返回的对数向量。x1 和 x2 用来控制端点，返回向量的两个端点的大小为 10^{x1} 和 10^{x2}。格式 1 中向量的长度是 50。格式 2 中的 n 用于指定向量的元素个数。函数 logspace 的计算过程是先得到 x1 和 x2 之间的等间距向量，然后计算 $10^{linspace(x1,x2,n)}$，因此函数 logspace 得到的向量不是等间距的，取对数后才是等间距的。

此外，还可以利用逗号和空格符来分开向量元素。

例 3–3： 用逗号和空格定义向量。

```
a1 = [1,2,3,4,1]
a2 = [5 3 4 1 1]
a3 = [4 5,1 3,2]
```

可见逗号和空格符可以混合使用。函数 randperm 可以用于产生一个元素为从 1 到 N 的随机自

然数序列，其调用格式如下：

```
rand('state',S)
v = randperm(N);
```

参数说明： S 用于设定随机数的状态，这样程序每次执行结果都是一样的，用户可以改变 S 的值以便选取不同的随机数。N 用来定义随机向量中的最大正整数。

例 3-4： 生成随机序列。

```
rand('state',22)
v = randperm(6)
```

上述程序将输出：

```
v =     1     5     6     2     3     4
```

向量元素的索引可以用下面的形式：v(n)调用行向量（或者列向量）的第 n 个元素，或者 v(n,1)调用列向量的第 n 个元素。v(n1:n2)和 v(n1:n2,1)表示向量 v 的第 n1 个元素和第 n2 个元素组成的向量，当 n1<n2 时，v(n1:n2)和 v(n1:n2,1)是一个空矩阵。此外，结果 v(s)可以调用 s 位置处的向量元素，这里 s 是一个由位置索引组成的向量。

3.2 向量运算函数

本节介绍与向量计算相关的一些 MATLAB 函数。

3.2.1 判断矩阵是否为向量

函数 isvector 可以用来判断一个矩阵是否为向量，其调用格式为：

```
r = isvector(v)
```

参数说明： v 可以是数、向量以及矩阵。r 是返回的判断结果。当 v 是数和向量（包含行向量和列向量）时，r 等于 1；当 v 是一个矩阵时，r 等于 0。这里 v 可以是 double 型数据、字符串以及符号变量。

例 3-5： 函数 isvector 的应用。

```
A=magic(4);
B=[1,1,2,2,3,3];
c1=isvector(A)
c2=isvector(B)
```

输出为：

```
c1 =     0
c2 =     1
```



这里 A 是矩阵而 B 是向量，函数 isvector 分别用返回值 0 和 1 表示 A 和 B 是否为矩阵。

3.2.2 向量的长度

函数 length 可以用来计算向量的长度，其调用格式为：

```
L = length(x);
```

参数说明： L 为返回的向量长度。x 是输入的向量或者矩阵。当 x 是一个非空矩阵的时候，L 等于矩阵的行数和列数的最大值；如果 x 是空矩阵，L 等于 0。

例 3-6： length 函数的应用。

```
A=1:5;
B=[1,1,2;2,3,3];
c1=length(A)
c2=length(B)
```

输出为：

```
c1 =     5                              c2 =     3
```



这里 A 是一个含 5 个元素的向量，函数 length 返回了它的长度；而 B 是一个 2×3 的矩阵，此时 length 返回了行数和列数的最大值。

3.2.3 向量的外积

函数 cross 可以用来计算向量的外积，其调用格式为：

```
c = cross(a,b);
```

参数说明： c 是返回的外积结果。a 和 b 要求有相同的行数和列数，当它们的行数或者列数等于 3 时才可以进行计算。

例 3-7： 计算向量的外积。

```
c1 = cross(1:3,3:5)
rand('state',2);         % 设置随机数的状态
c2 = cross(rand(3,4),rand(3,4))
```

上述语句所得结果如下：

```
c1 =    -2     4    -2
c2 =    0.1424   -0.0651   -0.0858    0.1099
       -0.4693    0.0951    0.5239    0.1628
        0.0900    0.1895   -0.0513   -0.2941
```



这里 c1 是向量外积结果，c2 给出的是各列分别进行外积计算的结果。

3.2.4 向量的内积

函数 dot 可以用来计算向量的内积，其调用格式为：

```
c = dot(A,B);
```

```
c = dot(A,B,dim);
```

参数说明： c 是返回的内积结果。A 和 B 是输入的向量或者矩阵，它们要求有相同的行数和列数。dim 用于指定内积计算的维数。当 dim 等于 1 时对矩阵的列向量进行内积计算，dim 默认为 1；当 dim 等于 2 时，对矩阵的行向量进行计算。

例 3-8： 下面是调用函数 dot 的例子。

```
rand('state',12);  % 设置随机数的状态
A = rand(2,4);
B = rand(2,4);
c1 = dot(A,B)
c2 = dot(A,B,2)
```

输出结果为：

```
c1 =    0.7589    0.6528    0.0764    1.1862
c2 =    0.9828
        1.6915
```



c1 和 c2 分别是对列向量和行向量计算内积的结果。

3.2.5 求解线性趋势项

函数 detrend 用快速傅里叶算法（FFT）去除向量中的线性趋势项，其调用格式如下：

```
y = detrend(x);                   % 格式 1
y = detrend(x,'constant');        % 格式 2
y = detrend(x,'linear',bp);       % 格式 3
```

参数说明： y 是返回的结果。x 是输入的向量。在格式 2 中，constant 用于指定去除的量是向量的均值，当 x 是矩阵的时候，去除的是各个列向量的均值。在格式 3 中，可以去除连续分段线性趋势项。

例 3-9： 下面是求解线性趋势项的例子。

```
A = [1,2,5,4];
B1 = detrend(A)
B2 = detrend(A,'constant')
B3 = detrend(A,'linear',2)
```

输出结果为：

```
B1 =   -0.2000   -0.4000    1.4000   -0.8000
B2 =    -2    -1     2     1
B3 =    0.0000   -0.6667    1.3333   -0.6667
```



这里给出 3 种不同形式趋势项去除的结果：B1 是直接去除趋势项的结果；B2 是以常数 3 去除趋势项的结果；B3 是进行分段去除趋势项的结果。

3.2.6 反转向量顺序

函数 wrev 可以反转一个向量的顺序，其调用格式为：

```
y = wrev(x);
```

参数说明：y 是反转后的向量。x 是输入的向量。当 x 是一个向量时，x 和 y 的行数和列数相等；当 x 是一个矩阵时，y 是一个向量。

例 3-10：下面是调用函数 wrev 的例子。

```
y1 = wrev(1:4)
x = magic(3)
y2 = wrev(x)
```

输出如下：

```
y1 =     4     3     2     1
x =     8     1     6
        3     5     7
        4     9     2
y2 =     2     7     6     9     5     1     4     3     8
```

用户可以对比输入和输出的顺序来理解 wrev 函数的用法。可以用 sort 来排列向量中元素的大小顺序，输出结果是按从大到小或者从小到大的顺序，同时这个函数也可以对矩阵各行或者各列元素的大小顺序进行排列。该函数的调用格式为：

```
y = sort(x);
y = sort(x,dim);
[y,Ix] = sort(x,dim,mode);
```

参数说明：y 是输出的排序结果。Ix 用来记录排序前后的位置关系。x 是输入的向量或者矩阵。dim 用于指定对行向量或者列向量进行排序。当 dim=1 时，对列向量进行排序；当 dim=2 时，对行向量进行排序。参数 mode 用来指定输出结果的大小顺序。当 mode='descend'时，输出 y 中元素的顺序是从大到小；当 mode='ascend'时，输出 y 中元素是从小到大，此为 mode 参数的默认值。

例 3-11：下面是调用 sort 函数进行排序的例子。

```
a = [3,2,1,2,3];
A = magic(3);
y1 = sort(a)
[y2,I2] = sort(a,2,'descend')
[y3,I3] = sort(A,2)
[y4,I4] = sort(A,2,'descend')
```

输出结果如下：

```
y1 =     1     2     2     3     3
y2 =     3     3     2     2     1
I2 =     1     5     2     4     3
y3 =  1     6     8
      3     5     7
```

```
y4 =
       8     6     1
       7     5     3
   9     4     2
I4 =
```

```
      2    4    9                       1    3    2
I3 =  2    3    1                   3    2    1
      1    2    3                       2    1    3
      3    1    2
```



这里给出了对向量和矩阵进行排序的结果，其中 I2，I3 和 I4 是相应的排序序号。

对于 descend 模式，可以利用下面的形式进行等价转换，如 y = sort(a,2,'descend')可以另写为 y = fliplr(sort(a,2))，而 y = flipud(sort(a,1))表示在竖直方向从上到下按由小到大排列。

3.3 集合的定义及相关运算

本节介绍与集合运算相关的几个函数，包括常见的交集、差集、并集等。

3.3.1 集合的交集

函数 intersect 可以计算集合的交集，其调用格式为：

```
c1 = intersect(a,b);                          % 格式 1
c2 = intersect(a,b,'rows');                   % 格式 2
[c3, Ia, Ib] = intersect(a,b);                % 格式 3
[c4, Ia, Ib] = intersect(a,b, 'rows');        % 格式 4
```

参数说明：c1，c2，c3，c4 是返回的交集结果。a 和 b 是输入的向量或者矩阵。在格式 1 中，a 和 b 要求是向量，它们的元素个数可以不等。在格式 2 中，a 和 b 是矩阵，它们的列数要求相等但行数可以不等，此时寻找 a 和 b 相同的行向量。Ia 和 Ib 分别是向量中相同元素的位置或者是相同行向量的位置，用户可以用索引 a(Ia)，b(Ib)，a(Ia,:)，b(Ib,:)来调用这些相同的元素或者行向量。

例 3-12：下面的 4 个例子分别对应上面 4 种格式。

```
a = [1,2,3,4];     % 对输入向量赋值
b = [5,6,3,2];     % 对输入向量赋值
A = [1,2;3,4;3,6]; % 对输入矩阵赋值
B = [1,3;1,2;3,4]; % 对输入矩阵赋值
c1 = intersect(a,b) % 计算两个向量的交集
c2 = intersect(A,B,'rows') % 计算两个矩阵中行向量的交集
[c3,Ia1,Ib1] = intersect(a,b) % 返回交集结果和交集元素对应的位置
[c4,Ia2,Ib2] = intersect(A,B,'rows') % 返回交集结果和交集元素对应的位置
```

上述语句运行结果如下：

```
c1 =    2    3                  c4 =    1    2
c2 =    1    2                          3    4
        3    4                  Ia2 =   1
c3 =    2    3                          2
Ia1 =   2    3                  Ib2 =   2
Ib1 =   4    3                          3
```



读者可以比较交集结果和交集的位置以理解该函数的用法。

3.3.2 集合中元素的判断

函数 ismember 可以用来判断一个数或者行向量是否为集合中的元素，其调用格式为：

```
r = ismember(a,s)                      % 格式 1
r = ismember(a,s,'rows');              % 格式 2
```

参数说明： r 是返回的判断结果。在格式 1 中，a 是数，s 是一个向量或者矩阵。在格式 2 中，a 是一个向量或者矩阵，r 和 a 具有相同的行数和列数。如果向量或者矩阵 s 中存在 a 中的元素，那么对应输出为 1，否则为 0。

例 3–13： 调用函数 ismember。

```
a1 = 1;            % 对变量赋值
A = 1:4;           % 对变量赋值
M = magic(3)       % 生成一个矩阵
a2 = [10,0;8,4];   % 生成 2 行 2 列矩阵
r1 = ismember(a1,A)  % 判断是否为集合中的元素
r2 = ismember(a1,M)  % 判断是否为集合中的元素
r3 = ismember(a2,M)  % 判断是否为集合中的元素
```

执行结果如下：

```
M =     8     1     6                    r2 =     1
        3     5     7                    r3 =    0     0
        4     9     2                            1     1
r1 =     1
```



这里给出检查一个数是否为数组中的元素，以及检查小的矩阵中各个元素是否为数组元素的例子，从中读者可以体会 ismember 的用法。

3.3.3 两个集合的差集

函数 setdiff 可以用来计算两个集合的差集，其调用格式如下：

```
s = setdiff(a,b);                      % 格式 1
s = setdiff(a,b,'rows');               % 格式 2
[s, Ic] = setdiff(a,b);                % 格式 3
[s, Ic] = setdiff(a,b,'rows');         % 格式 4
```

参数说明： s 是返回的差集结果。A 和 B 是向量或者矩阵，它们的行数和列数可以不相等。

例 3–14： 调用函数 setdiff。

```
a = [1,2,3,4];
b = [4,5];
A = magic(3);
B = [1,2,3;3,5,7];
c = setdiff(a,b) % 计算差集
```

```
C = setdiff(A,B,'rows')   % 对行向量计算差集
```

输出结果如下：

```
c =     1     2     3
C =
     4     9     2
     8     1     6
```



c 和 C 分别是以元素和行向量为集合单元而进行的差集计算结果。

3.3.4 集合异或运算

函数 setxor 可以运算集合异或（不在交集中的元素），其调用格式如下：

```
x = setxor(a,b);                          % 格式 1
x = setxor(a,b,'rows');                   % 格式 2
[x,Ia,Ib] = setxor(a,b);                  % 格式 3
[x,Ia,Ib] = setxor(a,b,'rows');           % 格式 4
```

参数说明： x 表示返回的异或结果。a 和 b 是向量或者矩阵（在格式 1 和格式 3 中，a 和 b 是向量；在格式 2 和格式 4 中，a 和 b 是矩阵，且要求 a 和 b 具有相同的列数）。Ia 和 Ib 是向量元素位置或者矩阵行位置。

例 3–15： 调用 setxor 的例子。

```
a = 1:3;
b = 2:4;
A = [1,2;3,4;5,6];
B = [3,4;1,6];
x1 = setxor(a,b)        % 计算集合异或运算
x2 = setxor(A,B,'rows')   % 对行向量进行异或运算
[x3,Ia1,Ib1] = setxor(a,b)   % 计算集合异或运算，并记录位置
[x4,Ia2,Ib2] = setxor(A,B,'rows')   % 对行向量进行异或运算，并记录位置
```

上述程序输出如下：

```
x1 =     1     4
x2 =     1     2
         1     6
         5     6
x3 =     1     4
Ia1 =     1
Ib1 =     3
x4 =     1     2
         1     6
         5     6
Ia2 =
         1
         3
Ib2 =     2
```



上述例子给出了以元素和行向量为集合单元时相应的异或运算过程的演示。

3.3.5 集合的并集

函数 union 可以用来计算两个集合的并集，其调用格式如下：

```
u = union (a,b);                    % 格式 1
u = union (a,b,'rows');             % 格式 2
[u,Ia,Ib] = union (a,b);            % 格式 3
[u,Ia,Ib] = union (a,b,'rows');     % 格式 4
```

参数说明： u 表示返回的异或结果。a 和 b 是向量或者矩阵（在格式 1 和格式 3 中，a 和 b 是向量；在格式 2 和格式 4 中，a 和 b 是矩阵，且要求 a 和 b 具有相同的列数）。Ia 和 Ib 是向量元素位置或者矩阵行位置。

例 3-16： 调用 union。

```
a = 1:2;
b = 2:3;
A = [1,2;5,3];
B = [3,4;1,2;3,5];
[u1,Ia1,Ib1] = union(a,b) % 计算并集
[u2,Ia2,Ib2] = union(A,B,'rows') % 以行向量为单元计算并集
```

上述程序输出如下：

```
u1 =     1     2     3
Ia1 =     1
Ib1 =     1     2
u2 =   1     2
       3     4
       3     5
       5     3
Ia2 =     2
Ib2 =     2
          1
          3
```



上述例子给出了以元素和行向量为单元时相应的集合之间并集计算过程。

3.3.6 去除重复的元素

函数 unique 可以返回一个没有重复元素的向量（去掉其中相同的元素），同时这个函数也可以处理矩阵的行向量，其调用格式如下：

```
u = unique(a);              % 格式 1
u = unique(a,'rows');       % 格式 2
[u,m,n] = unique(...);      % 格式 3
```

参数说明： u 是返回的处理结果。当输入用参数 rows 指定时，将去掉相同的行向量，同时返回的 u 是一个矩阵。当 u 是一个向量的时候，u 中元素按从小到大的顺序排列；当 u 是按第一列元素从小到大排列时，如果第一列元素相等，将参考对应的第二列元素的大小关系。a 是输入的元素或者向量。当 a 是矩阵且不指定参数 rows 的时候，返回的 u 是一个列向量，并去掉了矩阵中相同的元素，此时相当于执行 u=unique(a(:))。参数 m 用于指定向量 u 中各个元素在向量 a 中的位置，这里给出的是 a 中最后出现该元素的位置。参数 n 用于指定向量 a 中各个元素在向量 u 中的位置。对于 a 是矩阵的情况，m 和 n 是指对应行向量的位置。

例 3-17： 调用 unique 函数。

```
a = [1,2;1,2,3];
A = [1,2;3,4;1,2];
```

```
u1 = unique(a) % 删去向量中的重复元素
u2 = unique(A,'rows')              % 删去矩阵中重复的行向量
[u3,m3,n3] = unique(a)             % 删去向量中的重复元素，并记录元素对应的位置
[u4,m4,n4] = unique(A,'rows')      % 删去矩阵中重复的行向量，并记录元素对应的位置
```

输出结果如下：

```
u1 =    1    2    3
u2 =    1    2
        3    4
u3 =    1    2    3
m3 =    3    4    5
n3 =    1    2    1    2    3
u4 =    1    2
        3    4
m4 =
     3
     2
n4 =
     1
     2
     1
```



这里以元素和行向量为单位进行去除重复集合单元删除的演示，读者可以借助例子理解函数 unique 的用法。

3.4 矩阵生成方法

前面介绍了向量的生成及相关的计算函数，本节来介绍矩阵的生成方法。在 MATLAB 中矩阵的生成可以通过直接赋值和由特殊函数生成矩阵等方法。

前面介绍了利用逗号和空格符可以分隔向量的元素，在矩阵的定义中，要用分号来分隔行向量。

例 3-18：矩阵生成的例子。

```
A = [1,2,3;4,5,6;7,8,9]
```

输出如下：

```
A =
     1     2     3
     4     5     6
     7     8     9
```

对于含有较多元素的矩阵，上述方法将是非常烦琐的。用户可以查看其中的规律，使用特殊函数或者循环结构来建立矩阵。

例 3-19：利用向量和矩阵来组成新的矩阵。

```
A1 = [1,2,3;4,5,6;7,8,9];
A2 = [0;0;0];
A3 = magic(3);
A4 = [0,0,0];
C1 = [A1,A2,A3]
C2 = [A1;A4]
```

输出如下：

```
C1 =
     1     2     3     0     8     1     6
     4     5     6     0     3     5     7
     7     8     9     0     4     9     2
C2 =
     1     2     3
     4     5     6
     7     8     9
     0     0     0
```

常用的矩阵元素索引方式如表 3.1 所示。

表 3.1 矩阵元素索引方式

语法格式	说明
A(m,n)	调用矩阵 A 中 m 行 n 列位置处的矩阵元素
A(m,:)	调用矩阵 A 中第 m 行所有元素
A(:,n)	调用矩阵 A 中第 n 列所有元素
A(m,n1:n2)	调用矩阵 A 中 m 行元素第 n1 列到第 n2 列之间的所有元素
A(m1:m2,n)	调用矩阵 A 中 n 列元素第 m1 行和第 m2 行之间的所有元素

其中 n1:n2 和 m1:m2 还可以用用户自己定义的向量 s 来替换，s 是一个由位置索引组成的向量。此外，读者可以通过格式 A(m, n:end)对第 m 行中第 n 列元素至最后一个元素进行操作。而 A(m, n:end-n0)也是合法的语句，表示第 m 行中从第 n 列开始至倒数第 n0+1 个元素。通过以上介绍的方法，读者就可以对矩阵进行定义以及相关的操作了。

3.5 特殊矩阵的生成

在 MATLAB 中可以用语句 “A = [ ];” 来定义一个空矩阵。函数 meshgrid 可以用向量生成新矩阵的各行各列，其调用格式为：

```
[X,Y] = meshgrid(x);
[X,Y] = meshgrid(x,y);
```

参数说明： x 和 y 是向量。X 和 Y 是生成的矩阵。

例 3-20： 下面是调用该函数的例子。

```
x=1:3;
y=1:4;
[X1,Y1] = meshgrid(x)
[X2,Y2] = meshgrid(x,y)
```

输出如下：

```
X1 =
     1     2     3
     1     2     3
     1     2     3
Y1 =
     1     1     1
     2     2     2
     3     3     3
```

```
X2 =
     1     2     3
     1     2     3
     1     2     3
     1     2     3
Y2 =
     1     1     1
     2     2     2
     3     3     3
     4     4     4
```

可见 X1 和 X2 的各行都一样，Y1 和 Y2 的各列都一样。因此函数 meshgrid 可以用于三维绘图中 XY 坐标的生成，此外也可以用于一些二元函数的定义而省去循环结构。一些特殊矩阵生成函数的用法如表 3.2 所示。

表 3.2 特殊矩阵生成函数表

调用格式	说明
ones(n), ones(m,n), ones(m,n,p,...)	生成全 1 矩阵或者高维数组
zeros(n), zeros(m,n), zeros(m,n,p,...)	生成全 0 矩阵或者高维数组
eye(n), eye(m,n)	生成单位矩阵，m 和 n 不等时，主对角元素等于 1
compan(p)	返回向量的伴随矩阵
[out1,out2,...] = gallery(matname, p1, p2, ...)	生成测试矩阵
hadamard(n)	生成哈达马得矩阵
hankel(C), hankel(C,R)	生成汉克尔矩阵
hilb(n)	生成希尔波特矩阵
invhilb(n)	生成逆希尔波特矩阵
magic(n)	生成魔方阵
pascal(n)	生成帕斯卡矩阵
toeplitz(C,R), toeplitz(R)	生成托普利茨矩阵
wilkinson(n)	生成维尔金森特征值测试矩阵
vander(v)	生成范德蒙矩阵

3.6 矩阵计算的基本函数

本节介绍与矩阵密切相关的函数，如行数和列数的计算、整形、对角矩阵、旋转和移动等。下面具体介绍这些内容。

3.6.1 大小及索引问题

本小节介绍矩阵大小计算和索引方面的知识。有了这些知识，我们就可以更好地对矩阵进行相关计算了。

3.6.1.1 求解矩阵的行数和列数

函数 size 可以用来计算矩阵的行数和列数，其调用格式如下：

```
D = size(A)
[D1,D2] = size(A)
Dn = size(A,dim)
```

参数说明： D 是返回的一个行向量，表示矩阵的行数和列数数据。D1 和 D2 分别是矩阵的行数和列数。Dn 是指定维数 dim 对应的数据，当 dim=1 时返回行数，当 dim=2 时返回列数。

例 3-21： 求解矩阵的行数和列数。

```
A=[1,2,4;6,1,2];
[m,n]=size(A)
m=size(A,1)
n=size(A,2)
```

输出为：

```
m =     2
n =     3
m =     2
```

```
n =      3
```



这里 m 和 n 分别是矩阵 A 的行数和列数。

3.6.1.2 矩阵所有元素的个数

函数 numel 可以用来计算矩阵所有元素的个数，其调用格式如下：

```
N = numel(A);
```

参数说明： N 是返回的元素总数。A 是输入的矩阵。语句 numel(A)和 prod(size(A))等价。当 A 是向量时，numel 的作用和 length 一样。

例 3-22： 计算矩阵所有元素的个数。

```
A=[1,2,4;6,1,2];
N=numel(A)
```

输出为：

```
N =      6
```



返回 N 的数值正好等于矩阵 A 中元素的总数。

3.6.1.3 不同索引格式的转换

函数 ind2sub 和 sub2ind 可以在单一索引和脚标索引之间转化，它们的调用格式为：

```
[m,n] = ind2sub(siz,Ind);
Ind = sub2ind(siz,m,n);
```

参数说明： m 和 n 分别指行和列位置对应的向量。siz 是由矩阵行数和列数组成的 1×2 的向量。Ind 是由单一索引组成的向量。

例 3-23： 矩阵 M=magic(3)的不同索引方式。

```
M =
     8     1     6
     3     5     7
     4     9     2
```

通过 M(2,3)和 M(8)用户都可以得到元素 7。

例 3-24： 通过函数 ind2sub 和 sub2ind 实现两种索引方式的转化。

```
[m,n] = ind2sub([3,3],[7,8])
ind = sub2ind([3,3],[1,2],[3,3])
```

输出如下：

```
m =     1     2                n =     3     3              ind =     7     8
```

通过下面的例子来说明这两种索引方式的差别。

```
M1 = zeros(4);
M2 = zeros(4);
m = [1:2,4];
n = [1:2,4];
Ind = sub2ind([4,4],m,n);
M1(m,n)=1
M2(Ind)=1
```

输出如下：

```
M1 =
     1     1     0     1
     1     1     0     1
     0     0     0     0
     1     1     0     1

M2 =
     1     0     0     0
     0     1     0     0
     0     0     0     0
     0     0     0     1
```

可见这两种索引方式得到了不同的结果。M1 中指定行列的所有元素都变成 1，而 M2 中只是主对角上的 3 个元素变为 1。这个结果用户在使用过程中需要注意一下，不要混用。

3.6.2 矩阵整形

有时为了某种需要，需要改变矩阵的行数和列数。本小节就来具体介绍几种函数改变矩阵的行数和列数。

3.6.2.1 改变矩阵的行数和列数

函数 reshape 可以用来改变矩阵的行数和列数，但是不改变矩阵的元素值和数目，其调用格式如下：

```
Y = reshape(X,M,N);            % 格式 1
Y = reshape(X,M,N,P,...);      % 格式 2
```

参数说明：Y 是整形后的矩阵。X 是输入矩阵。在格式 1 中，M 和 N 分别是矩阵的行数和列数。利用格式 2 用户可以把 X 整形为高维数组。

例 3-25：调用函数 reshape 的例子。

```
X = [1,2,3;4,5,6;7,8,9;10,11,12]
Y = reshape(X,2,6)
```

输出如下：

```
X =
     1     2     3
     4     5     6
     7     8     9
    10    11    12

Y =
     1     7     2     8     3     9
     4    10     5    11     6    12
```

对比 X 和 Y 的数字对应关系，可以发现 reshape 函数是逐列把 X 中的元素摆放到矩阵 Y 中。这个顺序在使用中用户需要注意，否则会引起不必要的麻烦。



B = reshape(A,N,1)可以简单地写为 B=A(:)。

3.6.2.2 连接两个矩阵

函数 cat 可以用来连接两个矩阵，但不改变矩阵元素的相对位置和数值，其调用格式为：

```
C = cat(dim,A,B);
```

参数说明：C 是组合的矩阵。A 和 B 是输入矩阵。dim 用于指定行或者列。当 dim=1 时，cat(1,A,B)等价于[A;B]，此时要求 A 和 B 具有相同的列数，可用函数 vertcat 来实现；当 dim=2 时，cat(2,A,B)等价于[A,B]，此时要求 A 和 B 具有相同的行数，可用函数 horzcat 来实现。

例 3-26：连接两个矩阵。

```
A = magic(2)
B = [1,2;3,4]
C1 = cat(1,A,B)
C2 = cat(2,A,B)
```

输出如下：

```
A =
     1     3
     4     2
B =
     1     2
     3     4
C1 =
     1     3
     4     2
     1     2
     3     4
C2 =
     1     3     1     2
     4     2     3     4
```

3.6.2.3 复制数组

函数 repmat 可以用来复制数组，用一个矩阵作为单元进行展开，其调用格式为

```
B = repmat(A,M,N);
```

参数说明：B 是输出的矩阵。A 是输入的矩阵，作为复制单元。M 表示矩阵在水平方向复制矩阵 A 的份数。N 表示在竖直方向复制矩阵 A 的份数。特别地，利用 repmat 函数把一个行向量在竖直方向复制 N 份，把一个列向量水平地赋值 M 份，所得的两个结果可以和 3.5 节介绍的 meshgrid 函数的结构相同。

例 3-27：调用 repmat 函数来复制数组。

```
repmat(magic(2),2,3)
repmat(uint8(5),2,3)
```

输出如下：

```
ans =
     1     3     1     3     1     3
     4     2     4     2     4     2
ans =
```

```
1     3     1     3     1     3              5     5     5
4     2     4     2     4     2              5     5     5
```

3.6.3 对角矩阵

函数 diag 可以利用向量生成一个对角矩阵，其调用格式为：

```
A = diag(v);
A = diag(v,k);
```

参数说明： A 是返回的矩阵。v 是一个向量。k 是一个整数。当 k 是正数时，向量 v 被安排在主对角线上面第 k 条斜线上；当 k 为负数时，向量 v 被安排在主对角线下面第-k 条斜线上。返回矩阵是一个方矩阵，行元素个数为 abs(k)+length(v)。利用这个函数定义矩阵时，可以简化用户输入的内容。

例 3-28： 调用函数 diag 的例子。

```
v = 1:3;
a1 = diag(v)
a2 = diag(v,2)
a3 = diag(v,-2)
```

输出如下：

```
a1 =
     1     0     0
     0     2     0
     0     0     3
a2 =
     0     0     1     0     0
     0     0     0     2     0
     0     0     0     0     3
     0     0     0     0     0
     0     0     0     0     0
a3 =
     0     0     0     0     0
     0     0     0     0     0
     1     0     0     0     0
     0     2     0     0     0
     0     0     3     0     0
```



从这个例子，读者可以看出函数 diag 可以把向量 v 放置到主对角线上或者和主对角线平行的位置上。

3.6.4 矩阵旋转与移动

本节介绍矩阵的旋转和移动操作，这些操作可以方便用户在一些特殊场合来简化编程。

3.6.4.1 矩阵的转置

函数 transpose 可以用来计算矩阵的转置，其调用格式为：

```
B = transpose(A);
```

参数说明： B 是输出的转置矩阵。A 是输入矩阵。当 A 是实数矩阵时，语句“B = transpose(A);”和“B = A';”是等价的；当 A 是复数矩阵时，“B = A';”在计算转置的同时进行复共轭计算，而“B = transpose(A);”仍然计算转置，相当于“B = A.';”。

例 3-29： 调用函数 transpose 的例子。

```
B = transpose([1+2i,2+1i;3+2i,4+3i])
```

输出如下：

```
B =
   1.0000 + 2.0000i   3.0000 + 2.0000i
   2.0000 + 1.0000i   4.0000 + 3.0000i
```

3.6.4.2 矩阵的旋转

函数 rot90 可以把矩阵按逆时针方向围绕矩阵中心旋转 90°，其调用格式为：

```
B = rot90(A);
B = rot90(A,K);
```

参数说明： B 是旋转后的矩阵。A 是输入矩阵。K 用来设定旋转角度，其为 90 的整数倍。

例 3-30： 下面是调用函数 rot90 的例子。

```
A = [1:3;4:6];
B1 = rot90(A)
B2 = rot90(A,450)
```

输出如下：

```
B1 =
     3     6
     2     5
     1     4
B2 =
     6     5     4
     3     2     1
```



读者可以根据例子的结果理解函数 rot90 对矩阵操作后矩阵中各个元素位置变化的规律。

3.6.4.3 翻转矩阵

函数 fliplr 和 flipud 可以分别实现左右和上下翻转矩阵，其调用格式为：

```
Y1 = fliplr(X);
Y2 = flipud(X);
```

参数说明： Y1 和 Y2 是翻转后的矩阵。X 是输入矩阵。类似地，用户还可以用 flipdim 函数来翻转矩阵。

例 3-31： 下面是调用函数 fliplr 和 flipud 的例子。

```
X = [1:3;4:6];
Y1 = fliplr(X)
Y2 = flipud(X)
```

输出如下：

```
Y1 =
     3     2     1
     6     5     4
Y2 =
     4     5     6
     1     2     3
```



读者可以在水平方向上比较 X 和 Y1 各元素的对应关系，在竖直方向上比较 X 和 Y2 各元素的对应关系，从而了解函数 fliplr 和 flipud 的作用。

3.6.4.4 矩阵元素的移动

函数 wshift 可以用来实现矩阵元素在水平方向和竖直方向上的移动，其调用格式如下：

```
Y = wshift(type,X,P);
```

参数说明： Y 是移动后的矩阵。type 是一个字符串，其取值可以为'1d'或者'1D'。X 是输入矩阵。P 用来定义移动的量。

例 3-32： 下面是调用函数 wshift 的例子。

```
x = 1:6
X = magic(4)
y = wshift('1D',x,2)
Y = wshift('2D',X,[-1 8])
```

输出结果如下：

```
x =     1     2     3     4     5     6      y =     3     4     5     6     1     2
X =                                          Y =
   16     2     3    13                           4    14    15     1
    5    11    10     8                          16     2     3    13
    9     7     6    12                           5    11    10     8
    4    14    15     1                           9     7     6    12
```



读者在比较向量 x 和 y 的元素关系时可以查看元素的错位移动情况，而在比较矩阵 X 和 Y 的时候可以查看行向量位置变化关系。

3.6.4.5 用 circshift 来移动矩阵元素

函数 circshift 同样可以用来在水平和竖直方向上移动元素，其调用格式为

```
B = circshift(A,shiftsize);
```

参数说明： B 是移动元素后的矩阵。A 是输入矩阵。shiftsize 是一个数或者 1×2 的向量，其作用相当于 wshift 函数中的参数 p。

例 3-33： 调用 circshift 函数。

```
A = [ 1 2 3;4 5 6; 7 8 9];
B1 = circshift(A,1)
```

```
B2 = circshift(A,[-1,1])
```

输出如下：

```
B1 = 
     7     8     9
     1     2     3
     4     5     6
B2 =
     6     4     5
     9     7     8
     3     1     2
```



读者可以在竖直方向上比较矩阵 A 和 B1 行向量的位置关系，而在比较 A 和 B2 各元素位置关系时可同时在水平和竖直方向上比较元素的错位移动情况。

3.6.5 矩阵大小的增减

本节介绍矩阵的元素增减的办法。可以通过把两个或者多个矩阵合并起来从而达到元素增加，或利用设置矩阵中某些元素为空矩阵的方法达到删除矩阵元素的目的。本节来介绍相关的增减矩阵元素的方法。

矩阵元素的增加，可以通过矩阵合并操作来完成，如：

```
C = [A;B];
C = [A,B];
```

执行这样的合并操作时需要注意矩阵的行数和列数的匹配。

还可以通过循环结果对矩阵进行增加元素的操作，如下面的语句：

```
A = magic(3)
A(1,4)=1
```

上面两条语句执行后输出如下：

```
A =
     8     1     6
     3     5     7
     4     9     2
A =
     8     1     6     1
     3     5     7     0
     4     9     2     0
```

上述结果中，对 A(1,4)赋值后把相应列向量的其他位置的元素赋值为 0。用户可以在后续程序中更换这些 0 元素。

用户可以整行或者整列地删除矩阵的元素，比如：

```
A(m,:) = [];              % 删除 m 行，或者可以指定 m 为多个行位置，如 m = [1,2,4,5];
A(:,n) = [];              % 删除 n 列，或者可以指定 n 为多个列位置，如 n = [2,4,6];
A(m1:m2,:) = [];          % 删除从 m1 行到 m2 行的内容
A(:,n1:n2) = [];          % 删除从 n1 列到 n2 列的内容
```

MATLAB 不支持非阵型的数组形式，用户无法从矩阵中单独删除一个元素，比如

```
M=[8 1 6;3 5 7 ;4 9 2 ]
```

用户想删除小于 5 的元素，而去掉这些元素后，剩下的内容不成为一个矩阵。当遇到需要删除矩阵中元素的任务时，可以把要删除的元素设为定义范围外的数值，比如矩阵元素定义范围是正数，那么可以把要删除的元素设为 0、负数、复数、NaN 等。把上面矩阵 M 中小于 5 的元素设为 0 可以用下面的方法：

```
M(M<5)=0
```

输出为：

```
M =
     8     0     6
     0     5     7
     0     9     0
```

再比如在画图的时候，用户可以把要删除的数设为 NaN，再用 plot 等函数绘图时，遇到 NaN 数值时，MATLAB 不进行绘图。

3.6.6 矩阵的本征值

本节介绍和矩阵本征值与本征向量相关的函数的用法。

3.6.6.1 函数 eig

利用函数 eig 可以求矩阵的本征值和本征向量，其调用格式为：

```
E = eig(X);
[V,D] = eig(X);
```

参数说明： E 是由本征值组成的列向量。X 是输入的方矩阵。V 是由本征向量组成的矩阵，其中的各个列向量是本征向量。D 是由本征值组成的矩阵，对角元素为本征值。这里本征值和本征向量是对应的。

例 3-34： 下面是调用函数 eig 的例子。

```
M = magic(3);
E = eig(M)
[V,D] = eig(M)
```

输出如下：

```
E =
   15.0000
    4.8990
   -4.8990
V =
   -0.5774   -0.8131   -0.3416
   -0.5774    0.4714   -0.4714
   -0.5774    0.3416    0.8131
```

```
D =
   15.0000         0         0
         0    4.8990         0
         0         0   -4.8990
```

3.6.6.2 矩阵行列式值的计算

函数 det 可以用来计算矩阵的行列式，其调用格式为：

```
D = det(X);
```

参数说明： D 是行列式的值。X 是输入的方矩阵。

例 3-35： 下面是一个求本征值的例子。

```
D=det(magic(3))
```

输出为：

```
D =
  -360
```

3.6.6.3 求解矩阵的秩

函数 rank 可以用来计算矩阵的秩，其调用格式为：

```
R = rank(X);
```

参数说明： R 是矩阵的秩。X 是输入的矩阵。

例 3-36： 下面是调用函数 rank 的例子。

```
R = rank(magic(4))
```

输出为：

```
R =     3
```

3.7 高维数组

除了前面介绍的矩阵，MATLAB 还支持高维数组。一般情况下，高维数组用于数据的存储，如彩色图片的数据利用一个 M×N×3 数组存储。虽然很多函数不支持高维数组的运算，但当需要进行一些计算的时候，可以把高维数组分解为矩阵进行处理。下面介绍几个和高维数组相关的函数。

3.7.1 计算数组维数

函数 ndims 可以用来计算数组维数，其调用格式为：

```
N = ndims(X);
```

参数说明： N 是返回的维数。X 是输入矩阵。

例 3-37： 下面是调用 ndims 的例子。

```
n1 = ndims(magic(3))
n2 = ndims(rand(2,3,4,5))
```

输出如下：

```
n1 =     2                                   n2 =     4
```

3.7.2 删除单独的维数

函数 squeeze 可以用来删除单独的维数，其调用格式为：

```
B = squeeze(A);
```

参数说明： B 是输出后的数组。A 是输入的数组。

例 3-38： 下面是调用函数 squeeze 的例子。

```
B = squeeze(rand(2,1,3))
```

输出如下：

```
B =
0.8339    0.9127    0.8335
0.9099    0.5087    0.0505
```

输出的 B 是一个 2×3 的矩阵。

3.7.3 移动数组维的顺序

函数 shiftdim 可以用来移动数组维的顺序，其调用格式为：

```
[Y,nshifts] = shiftdim(X);
Y = shiftdim(X,N);
```

参数说明： Y 是移动后的矩阵。nshifts 是移动后数组的维数。X 是输入的数组名。N 是指定移动的维数。

例 3-39： 下面是调用 shiftdim 函数的例子。

```
a = rand(1,1,3,1,2);
[b,n] = shiftdim(a);
c = shiftdim(b,-n);
d = shiftdim(a,3);
```

其中，b 是一个 3×1×2 的数组，n=2，c=a，d 是一个 1×2×1×1×3 的数组。输出如下：

```
a(:,:,1,1,1) =    0.2785
a(:,:,2,1,1) =    0.5469
a(:,:,3,1,1) =    0.9575
a(:,:,1,1,2) =    0.9649
a(:,:,2,1,2) =    0.1576
a(:,:,3,1,2) =    0.9706
b(:,:,1) =
   0.2785
   0.5469
   0.9575
b(:,:,2) =
   0.9649
   0.1576
   0.9706
n =    2
```

```
c(:,:,1,1,1) =    0.2785
c(:,:,2,1,1) =    0.5469
c(:,:,3,1,1) =    0.9575
c(:,:,1,1,2) =    0.9649
c(:,:,2,1,2) =    0.1576
c(:,:,3,1,2) =    0.9706
d(:,:,1,1,1) =    0.2785    0.9649
d(:,:,1,1,2) =    0.5469    0.1576
d(:,:,1,1,3) =    0.9575    0.9706
```



读者可以通过比较 a，b，c 和 d 对应元素位置的变化来了解函数 shiftdim 的作用。

3.7.4 改变数组的维数

函数 permute 可以用来改变数组的维数，其调用格式如下：

```
B = permute(A,order);
```

参数说明： B 是改变后的数组结果。A 是输入高维数组。order 用来指定改变数组的顺序，其值是一个向量，元素为 1~N（N 为数组 A 的维数）。函数 ipermute 和 permute 之间构成一个广义的转置矩阵的关系。

例 3-40： 调用函数 permute 的例子。

```
a = rand(2,3,4,5);
size(permute(a,[3 2 1 4]))
```

输出为：

```
ans =     4     3     2     5
```

3.7.5 计算高维函数的离散形式

函数 ndgrid 可以生成多维网格坐标来计算高维函数的离散形式，作用类似于前面介绍的 meshgrid 函数。该函数的调用格式如下：

```
[X1,X2,X3,...] = ndgrid(x1,x2,x3,...);
```

参数说明： X1，X2，X3 等是输出的高维网格。x1，x2，x3 等是对应坐标的采样点。这里输出的数组维数等于 length(x1) × length(x2) × length(x3) × …。

例 3-41： 调用函数 ndgrid 的例子。

```
[x1,x2,x3] = ndgrid(-2:.2:2, -2:.25:2, -2:.16:2);     % 生成三维网格矩阵
z = x2 .* exp(-x1.^2 - x2.^2 - x3.^2);                % 计算相应的函数表达式
slice(x2,x1,x3,z,[-1.2 .8 2],2,[-2 -.2])              % 绘制切片图
```

上述程序将得到一个切片图（如图 3.1 所示）。其中，x1，x2 和 x3 的 size 都等于 21×17×26。通过 x=-1.2，x=0.8，x=2，y=2，z=-2 和 z=-0.2 等平面把空间分为 6 个部分。

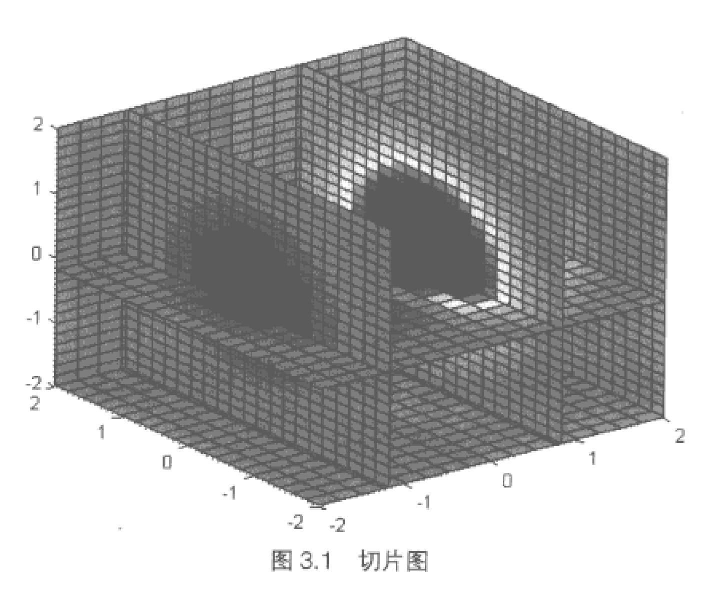


图 3.1　切片图

3.8 小结

本章主要介绍了向量和矩阵的定义和相关操作函数。很好地利用向量和矩阵的运算，可以帮助用户建立快速而简洁的程序。首先介绍了向量的定义方法，以及相关的向量运算函数和操作函数；然后介绍了 6 个集合运算函数的用法，使用这些函数用户可以完成常见的集合运算。矩阵是 MATLAB 的一个核心元素，MATLAB 提供了大量支持矩阵计算的函数。这里先给出了矩阵的定义方法以及一些产生特殊矩阵的函数；接下来详细介绍了与矩阵相关的函数，包括行数和列数计算、索引方法、整形、对角矩阵、旋转操作、平移操作、矩阵元素的增减、本征值等方面的知识。最后介绍了高维数组及相关操作的函数。

◆　　◆　　◆

第4章 表 达 式

本章包括

- ◆ **算术表达式** 介绍算术表达式的建立和相关知识。
- ◆ **关系表达式** 介绍关系运算符的用法。
- ◆ **逻辑运算** 给出相关的逻辑运算函数的用法。
- ◆ **符号计算** 介绍符号变量的建立以及如何进行一些符号计算。
- ◆ **特殊函数与特殊多项式** 介绍如何计算一些特殊函数，其中包括多项式。
- ◆ **卷积与相关** 介绍卷积和相关这两种运算。
- ◆ **使用中的一些技巧** 列举了一些函数的使用技巧。

表达式是程序语句的基本单元。本章主要介绍各类表达式的相关知识，如算术表达式、关系表达式和逻辑表达式等。尽管这些表达式和其他高级编程语言相似，但是 MATLAB 大部分操作都是以矩阵作为计算对象的基本单元，很多运算符都是针对矩阵的，这一点与其他语言不大相同。矩阵化的程序风格是 MATLAB 的一大特色，用户在利用这些运算符号时需要给予足够重视。本章还介绍一些多项式函数相关操作的例子，同时给出特殊函数的计算方法。

4.1 算术表达式

算术表达式是在编程工作中经常用到的，在表 4.1 中，总结了相关算术符号的含义及使用时需要注意的事项。

表 4.1　算术符号说明

符号	功能	注意事项
+	矩阵加法	要求 2 个矩阵的 size 相同
-	矩阵减法	要求 2 个矩阵的 size 相同
*	矩阵乘法	要求第 1 个矩阵的列数等于第 2 个矩阵的行数
/	矩阵右除法	要求 2 个矩阵的列数相同
\	矩阵左除法	要求 2 个矩阵的行数相同*
.*	矩阵元素之间的乘法	要求 2 个矩阵的 size 相同
./	矩阵元素之间的右除法	要求 2 个矩阵的 size 相同
.\	矩阵元素之间的左除法	要求 2 个矩阵的 size 相同**
^	矩阵的幂运算	要求矩阵是方矩阵
.^	矩阵元素的幂运算	无要求



* A/B 和[B'\A']'是相等的，这个关系对于单个数字也成立。** A./B 和 B.\A 是相等的，但是实际应用中大多数用户采用 A./B 形式来计算矩阵元素除法。

在计算一些函数的离散形式的函数值时，表 4.1 中描述的矩阵元素的乘除法往往可以方便地来

计算，从而使表达式更为简单。比如，计算二元函数 $f(x,y)=xy+x^2+y^3$（x 和 y 定义在[-3,3]区间内）离散情况下的值，可以利用下面的语句来完成：

```
[x,y] = meshgrid(linspace(-3,3,200));      % 对x和y离散取点
f = x.*y+x.^2+y.^3;
```

这样就可以计算出函数 $f(x,y)$ 的离散值。

Kronecker 张量积的计算在 MATLAB 中可以调用函数 kron 来实现，其调用格式为：

```
K = kron(X, Y);
```

参数说明： X 和 Y 是两个矩阵，矩阵 K 的 size 可以按 size(K)=size(X).*size(Y)计算。假设 X 是一个 2×3 的矩阵，那么这 3 个矩阵之间元素的关系用 MATLAB 语言可以表示为：

```
K = [X(1,1)*Y, X(1,2)*Y, X(1,3)*Y;
X(2,1)*Y,X(2,2)*Y,X(2,3)*Y];
```

当 X 或者 Y 是稀疏矩阵的时候，只有非零元素参与计算，同时矩阵 K 将是稀疏矩阵。

例 4-1： Kronecker 张量积的计算。

```
x=[2,3;1,4];
y=[1,2,3;4,5,6];
K=kron(x,y)
```

输出为：

```
K =
     2     4     6     3     6     9
     8    10    12    12    15    18
     1     2     3     4     8    12
     4     5     6    16    20    24
```



在编程的时候可以对比前面的介绍，理解 K 与输入参数的关系。

4.2 关系表达式

关系运算符号可以用来对数和矩阵对象进行大小关系比较，同时返回比较结果（由 0 和 1 组成的矩阵），这些结果用户可以应用于条件语句或者一些表达式中。基本的关系运算符号有>，<，>=，<=，= =，~=等 6 个，其中后 4 个关系运算符号是由两个符号组成的，用户在调用这 4 个符号时中间不能有空格，顺序也不能改变。在关系运算符两侧写入的矩阵要求具有相同的 size。关系运算符将对矩阵对应元素进行比较，如果对应元素的大小关系和关系运算符的含义相同，那么返回矩阵的值将等于 1，反之则等于 0。矩阵数据类型可以是 double 型或字符型的数据。

下面以 A = [1, 2, 3]和 B=[2, 2, 2] 为例，说明这 6 个运算符返回的结果。令 C = A**B，这里**代表上面提到的 6 个关系运算符，该表达式的含义是把 A**B 的计算结果赋值给 C，C 的结果如表 4.2 所示。

表 4.2 关系运算符举例

关系运算符	>	<	>=	<=	==	~=
C	[0, 0, 1]	[1, 0, 0]	[0, 1, 1]	[1, 1, 0]	[0, 1, 0]	[1, 0, 1]

其中，C 的数据类型是 logical，它可以和 double 型数据进行进一步的计算。

4.3 逻辑运算

逻辑运算和逻辑函数在很多计算语言中都扮演着重要的角色，它们可以针对逻辑型数据进行处理。在 MATLAB 中对逻辑运算和逻辑函数的功能又进行了扩展，其中包含 "与"、"或"、"非" 和 "异或" 4 种基本的逻辑运算，用 0 表示 "假"，而任意非 0 值都表示 "真"（这样，逻辑运算就可以支持逻辑型变量和 double 型变量）。MATLAB 的逻辑运算也是支持矩阵计算的。

4.3.1 基本运算

符号 "&"、"|" 和 "~" 分别代表基本逻辑运算关系 "与"、"或" 和 "非"，而函数 xor 可以用来计算 "异或"。使用这 4 个逻辑运算时需要注意以下 3 点：

- 数和矩阵进行逻辑运算时，数将和矩阵的所有元素进行运算。
- 在逻辑运算符、关系运算符和算术运算符之间，逻辑运算符的优先级最小，但是 "非" 逻辑运算的优先级最高，因此用户在编写程序的时候需要利用圆括号或者方括号来指定计算顺序。
- "与" 和 "或" 的优先级相同，执行顺序按从左至右。

除了上述基本逻辑运算之外，MATLAB 还提供了几个逻辑函数供用户在交互运算和矩阵处理中应用，利用它们可以方便地查找和更替矩阵中满足指定条件的所有元素。利用好这些函数，可以使程序变得简洁，有的时候可以增加程序的执行速度。下面将详细介绍这些函数的用法。

4.3.1.1 查找矩阵中非零元素的位置

函数 all 可以用来查找矩阵中非零元素的位置，如果指定位置的数非零，那么 all 函数将返回数值 1，否则为 0。其调用格式如下：

```
P = all(a);
P = all(A);
P = all(A,dim);
```

参数说明： P 是返回计算位置处非零数的信息。a 是向量。A 是矩阵。dim 指定相应的维数。当 dim=1 时，all 函数返回矩阵第一行非零数信息；当 dim=2 时，返回第一列非零数信息；当 dim=3 时，返回整个矩阵非零数信息。

例 4-2： 查找矩阵中非零元素的位置。

```
A=[1,2,0;0,2,3];
p1=all(A)
p2=all(A,2)
```

输出为：

```
p1 =                                    p2 =
     0     1     0                            0
                                              0
```



从这个例子可以看出，仅当对应的行向量（或者列向量）的所有元素都不等于 0 时，函数 all 对应的输出才等于 1。

4.3.1.2 判断矩阵中的列向量是否为 0

函数 any 可以用来判断矩阵中的列向量是否为 0，如果矩阵的列向量是非零向量，则返回 1，否则返回 0。其调用格式为：

```
p = any(a);
p = any(A);
p = any(A,dim);
```

参数说明： p 是返回计算位置处向量非零信息。a 是向量。A 是矩阵。dim 用于指定计算的维数。当 dim=1 时，返回各行向量非零情况；当 dim=2 时，返回各列向量非零情况；当 dim=3 时，对于矩阵计算，函数 any 和函数 all 作用一样，返回各元素非零情况。

例 4–3： 判断矩阵中的列向量是否为 0。

```
A=[1,0,0;0,2,0];
p1=any(A)
p2=any(A,2)
```

输出为：

```
p1 =                                    p2 =
     1     1     0                            1
                                              1
```



从这个例子可以看出，仅当对应的行向量（或者列向量）所有元素都为 0 时，函数 any 对应的输出才等于 0。

4.3.1.3 检测变量或者文件是否存在

函数 exist 可以用来检测变量或者文件是否存在，在防止重命名方面很有用，还可以根据变量或者文件存在的情况做下一步处理。该函数调用格式如下：

```
v = exist('A') ;
v = exist('A', 'var');
v = exist('A', 'builtin');
v = exist('A', 'file');
v = exist('A', 'dir');
v = exist('A', 'class');
```

参数说明： v 是返回的检测结果，是一个自然数。A 用于指定变量或者文件名称。对于不同 v 值对应 A 的含义如表 4.3 所示。

表 4.3 不同 v 值对应 A 的含义

v 的值	A 的属性	v 的值	A 的属性
0	变量或者文件不存在	5	A 是内建函数名
1	A 是变量名	6	p 码文件
2	M 文件或未知类型文件	7	A 是路径名
3	MEX 文件	8	A 是一个 Java 类
4	simulink 文件（MDL 文件）		

参数 var 用于指定检测变量。参数 builtin 用于指定检测内建函数。file 用于指定检测文件。dir 用于指定检测路径。class 用于指定 Java 类。

例 4-4：检测变量或者文件是否存在。

```
ABC=1;
P1=exist('Ch04.doc')
P2=exist('ABC')
```

输出为：

```
P1 =     2
P2 =     1
```



这个例子中，输出 P1.等于 2 表示对应的输入字符串是文件名，而 P2 等于 1 表示输入字符串是变量名。

4.3.1.4 得到非零元素的位置

前面介绍的 any 和 all 函数可以查找矩阵中的元素是否为非零元素，但是有的时候需要得到这些非零元素的位置。MATLAB 提供函数 find 来实现这一功能，利用这个函数用户可以方便地得到非零元素的位置，而不必对矩阵的所有元素逐一排查，在此基础上用户可以对查找出来的元素进行改动。该函数的调用格式为：

```
p = find(A);
[m, n] = find(A);
[m, n, v] = find(A);
```

参数说明：p 是矩阵 A 中非零元素的位置索引。m 和 n 是非零元素的行位置和列位置。v 是矩阵元素 A(m,n)的值。利用函数 find 还可以查找矩阵 A 中元素和数（或者同 size 矩阵）的大小关系比较，比如：

```
[m, n] = find(A>a) 或者 p = find(A>a) 得到矩阵 A 中所有大于 a 的元素的位置
[m, n] = find(A>B) 或者 p = find(A>B) 得到矩阵 A 大于矩阵 B 的元素的位置
```

对于其他关系运算符，可以根据上面的形式类似得到。用户应用函数 find 可以替代一些条件语句，从而使程序简化。

例 4-5：函数 find 的应用。

```
A = [1:3;3:5];
B = 1:6;
```

```
[am,an] = find(A>3)
bn = find(B==5)
```

输出为：

```
am =     2                              3
         2                    bn =      5
an =      2
```



这里输出 am，an 和 bn 表示满足条件的数据所在的位置，其中 am 表示元素所在的行位置，an 表示元素所在的列位置。

4.3.1.5 判断矩阵元素是否为有限值

函数 isfinite 可以用来确定矩阵元素是否是一个有限的数，是则在相应位置返回值为 1，否则返回 0。其中无限值有 NaN（或者 nan）和 inf，它们都被认为是无穷大。该函数调用格式如下：

```
p = isfinite(A);
```

参数说明： p 是一个和矩阵 A 同 size 的 logical 型矩阵，其相应位置元素的取值表示矩阵 A 是否为有限数的信息。此外，MATLAB 还提供了函数 isinf 和 isnan 来分别检测无穷大和非数（nan 或 NaN），其用法和函数 isfinite 相同，读者可以阅读帮助信息，这里不再赘述。

例 4-6： 确定矩阵元素是否为有限值。

```
A=[2,inf,-inf,nan];
p=isfinite(A)
```

输出为：

```
p =     1     0     0     0
```



这个例子表示在数组 A 中的无穷大（inf）和非数（NaN）的位置，函数 isfinite 将返回 0，其他数据的位置返回值等于 1。

4.3.1.6 检测空矩阵

函数 isempty 用来检测空矩阵，空矩阵的定义是矩阵存在但矩阵中没有元素。在 MATLAB 中可以定义空矩阵，一些语句未执行或者一些特殊语句（如 A = 9:1）也可以得到空矩阵。空矩阵在一些特殊运算时可能产生错误，因此如果在程序执行之前对是否可能成为空矩阵进行判断，可以很好地避免错误发生而影响程序的执行。该函数调用格式为：

```
p = isempty(A);
```

参数说明： A 是要判断的矩阵。如果 A 是空矩阵，则 p 等于 1，否则 p 为 0。这样我们可以根据输出参数 p 的取值情况确定矩阵 A 是否为空。

例 4-7： 检测空矩阵。

```
A=[];
```

```
B=[3,1,1];
p1=isempty(A)
p2=isempty(B)
```

输出为：

```
p1 =     1
p2 =     0
```



这个例子说明函数 isempty 可以检测矩阵是否为空，当是空矩阵时，函数 isempty 的返回值等于 1，反之等于 0。利用这个函数可以在程序段中判断矩阵是否为空，以避免一些错误出现而使程序连续进行下去。

4.3.1.7 判断多个变量是否相等

对于两个变量，我们知道可以利用关系符号“==”来判断，而函数 isequal 可以用来判断多个变量的数据类型、大小和值是否相等。该函数的调用格式如下：

```
p = isequal(A, B, C,…);
```

参数说明： A，B，C 等是变量名，可以是 double 型数据，也可以是字符串型数据。对于多个矩阵，如果所有对应位置的元素都相等，则函数 isequal 的返回值 p 等于 1，反之等于 0。

例 4-8： 判断多个变量的数据类型、大小和值是否相等。

```
A1=[1,2];B1=[1,2];C1=[1,2];A2=[1,1];
p1=isequal(A1,B1,C1)
p2=isequal(A1,B1,A2)
```

输出为：

```
p1 =     1
p2 =     0
```



这个例子表明仅当所有输入矩阵都相等时，函数 isqual 的返回值才等于 1，否则等于 0。

4.3.1.8 判断变量是否为数据型变量

函数 isnumeric 可以用来判断变量是否为数据型变量，以用来区分数据型变量和非数据型变量。该函数的调用格式为：

```
p = isnumeric(A);
```

参数说明： 当变量 A 的数据类型为 double，int8，uint8，int16，uint16，int32，uint32，int64 和 uint64 等数据类型时，p 等于 1，其他数据类型，如字符串、结构体、逻辑型数据，则 p 等于 0。类似地还可以使用函数 issparse 来区分稀疏矩阵，使用函数 isstr（以后可能被 ischar 替代）来区分字符串，使用函数 islogical 来区分逻辑型变量，使用函数 isobject 来区分 MATLAB 对象，使用函数 isjava 来区分 Java 对象矩阵，使用函数 isfield 来区分结构体矩阵地域，使用函数 isstruct 来

区分结构体，使用函数 iscell 来区分 cell 矩阵，使用函数 ishandle 来区分图形句柄。

例 4-9： 判断变量是否为数据型变量。

```
A1=[1.32,5.27];B1='AS';
p1=isnumeric(A1)
p2=isnumeric(B1)
```

输出为：

```
p1 =     1
p2 =     0
```



从这个例子可以看出，当输入内容为数值型数据时，函数 isnumeric 的返回值是 1，输入其他类型的内容时，返回值是 0。

4.3.2 腐蚀与膨胀运算

腐蚀运算和膨胀运算是数学形态学的两个基本操作，其中腐蚀操作是在对象边界处消除像素点，而膨胀操作则是腐蚀操作的逆过程，可以把它们看做是高级的逻辑运算函数，应用于图像处理问题中。

4.3.2.1 创建结构元素矩阵

在 MATLAB 中函数 strel 创建结构元素矩阵用于腐蚀和膨胀操作中，该函数的调用格式为：

```
se = strel(shape, parameters);
```

参数说明： se 是创建的结构元素矩阵，se 的数据类型是 strel。shape 是字符串，用于指定结构元素矩阵的形状。parameters 用于定义结构矩阵的参数，因形状不一，其格式也不一样。

例 4-10： 创建结构元素矩阵。

```
se1 = strel('square',6) ;         % 创建一个 6×6 的正方形
se2 = strel('line',7,45) ;        % 创建一条长度为 7 且角度为 45° 的斜线
se3 = strel('disk',8) ;           % 创建半径为 8 的圆盘
se4 = strel('ball',15,5) ;        % 创建一个半径为 15 且高度为 5 的椭圆体
```

4.3.2.2 计算腐蚀运算

函数 imerode 可以用来计算腐蚀运算，其调用格式为：

```
I2 = imerode(I1,se);
```

参数说明： I2 是腐蚀后的矩阵。I1 是原始矩阵。se 是结构元素矩阵，其由函数 strel 定义。

4.3.2.3 进行膨胀操作

函数 imdilate 可以用来进行膨胀操作，其调用格式如下：

```
I2 = imdilate(I1,se);
```

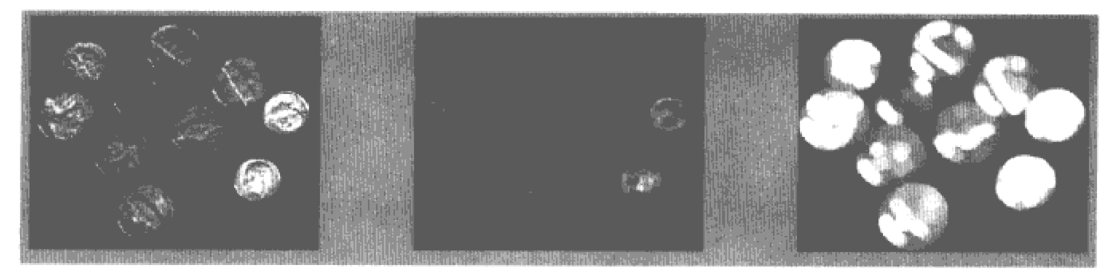
参数说明： I2 是膨胀后的矩阵。I1 是原始矩阵。se 是结构元素矩阵，由函数 strel 定义。

例 4–11： 下面是一个进行腐蚀和膨胀的例子。

对 MATLAB 图像工具箱中的 coins.png 文件对应的像素矩阵进行运算，相关程序如下：

```
A = imread('coins.png');                % 读入原始图像
se1 = strel('square',6);                % 生成结构元素矩阵 se1
se3 = strel('disk',8);                  % 生成结构元素矩阵 se3
E = imerode(A,se1);                     % 进行腐蚀运算
D = imdilate(A,se3);                    % 进行膨胀运算
subplot(131);imshow(A);                 % 绘图
xlabel('(a)','Fontsize',16);            % X 轴标注
subplot(132);imshow(E);                 % 绘图
xlabel('(b)','Fontsize',16);            % X 轴标注
subplot(133);imshow(D);                 % 绘图
xlabel('(c)','Fontsize',16);            % X 轴标注
```

上述程序执行后输出如图 4.1 所示的图形。



(a)原始图像　　(b)腐蚀的图像　　(c)膨胀的图像

图 4.1　腐蚀和膨胀的结果



图 4.1 给出了腐蚀和膨胀操作的效果图，其中图(b)中各个硬币的大小变小、亮度变暗，而图(c)中硬币大小变大、亮度变亮。

4.4　符号计算

利用 MATLAB 进行一些符号计算，可以处理简单的问题。而且借助于 maple 处理核，MATLAB 也具有较强的符号处理能力。本节将介绍如何利用 MATLAB 进行符号计算的一些方法。

4.4.1　变量的定义

在 MATLAB 中定义符号变量可以使用函数 sym 和 syms。此外，MATLAB 还提供了与符号变量操作相关的函数，本节来介绍这些函数的使用方法。

4.4.1.1　定义符号变量

函数 sym 可以用来定义符号变量，也可以用来直接定义符号表达式，其调用格式如下：

```
s = sym('name');
s = sym(var, flag);
```

参数说明： s 是返回的符号变量名（或者表达名式）。name 用来输入变量的名称或者表达式，

如 a，b，a+3*b+x^2。在变量个数不多时，建议用户把 s 和 name 写为相同的内容，这样在后续调用中不会引起混淆。当输入表达式的时候用户需要注意表达式的正确输入。var 是一个数值型变量名，它可以是数或者矩阵。flag 用来指定 var 的具体类型，flag 可以选择的参数有 f，r，e 和 d，其中 r 是默认值；f 表示浮点型数据，给出一个最接近 var 的浮点型数值结果；r 表示有理数；e 用来给出估计误差；d 表示十进制数。

例 4-12：定义符号变量。

```
a=sym('a')
b=5/3;
b=sym(b,'d')
```

输出为：

```
a = a
b =1.6666666666666667406815349750104
```



这个例子表明函数 sym 可以定义符号变量，同时也能把数值转化为符号变量。

4.4.1.2 特殊符号变量

与 sym 函数第二种调用格式作用类似的函数是 vpa，其调用格式为：

```
r = vpa(A);
r = vpa(A, D);
x = sym('x','positive');
y = sym('y','real');
z = sym('z','unreal');
```

参数说明：参数 A 是 double 型变量。D 是转化后小数总位数。输出结果 r 是一个符号串型数据。此外，用户还可以用函数 digits 来设定 D 的值，调用格式是 digits(N)，当 N 取小数时，MATLAB 将进行四舍五入取整。positive 用于指定符号变量是一个正数，real 用于指定符号变量是一个实数，unreal 用于指定符号变量是一个非实数。利用函数 sym 和 char 可以在符号变量和字符串型变量之间转化。

例 4-13：利用函数 vpa 实现转换。

```
d=9/7;
r=vpa(d,8)
```

输出为：

```
r =1.2857143
```



这个例子表明函数 vpa 可以把 double 型数据转化为带有指定数位的符号变量。

4.4.1.3 syms 定义符号变量

除了函数 sym 之外，用户还可以利用函数 syms 来定义符号变量，这个函数可以同时定义多个符号变量。在调用格式上它要比 sym 简单，其调用格式为：

```
syms arg1 arg2 ...
syms k positive
```

参数说明： arg1，arg2 等是定义的符号变量名，用户直接输入名称而不用字符串形式，这样函数 syms 定义多个符号变量时要比函数 sym 快捷得多。同样，用户可以用 positive，real 和 unreal 指定符号变量的属性。

4.4.1.4 查找符号变量

函数 findsym 可以用来从表达式中查找符号变量，其调用格式为：

```
F1 = findsym(S);
F2 = findsym(S, n);
```

参数说明： S 是一个符号表达式。F1 是表达式 S 中所有符号变量名、逗号和空格符组成的一个字符串，其中变量名是按英文字母顺序排列的，大写字母在小写字母之前。如果 S 中没有符号变量，F1 将是一个空的字符串，如 findsym(sym('1+sin(10)'))。F2 返回的是表达式 S 中最靠近字母 “x” 的 n 个符号变量。

例 4-14： 查找符号变量。

```
syms x
f=sym(['a+sqrt(c)+d^2'+x,'+X']);
F1=findsym(f)
```

计算得到 F1 的结果是：

```
F1 =X, a, c, d, x
```



这个例子表明函数 findsym 可以找出表达式中所有的变量名。

4.4.1.5 把符号表达式转为手写格式

利用函数 pretty 可以把符号表达式转为手写格式，在这个转化过程中 MATLAB 不对表达式进行化简和展开操作。该函数调用格式为：

```
pretty(S)
```

参数说明： S 是符号表达式或者符号表达式对应的变量名。

例 4-15： 符号表达式转为手写格式的形式。

```
pretty(sym('sqrt(x^2+y^2)'))
```

输出为：

```
  2    2 1/2
(x  + y )
```



该例子表明函数 pretty 可以把表达式显示为手写形式。

4.4.1.6 对符号表达式进行化简

利用函数 simple 可以对符号表达式进行化简操作。其调用格式为：

```
simple(S);
[R, how] = simple(S);
```

参数说明：S 是要化简的表达式。R 是返回的最简化的表达式。how 是一个字符串，指出所用的化简方法。

例 4-16：举例说明 simple 函数的用法，执行下面的语句对表达式进行化简。

```
simple(sym('sin(x)^2+sin(y)^2+(x+y)^3'))
```

运行之后输出结果如下：

```
simplify:
2-cos(x)^2-cos(y)^2+x^3+3*x^2*y+3*x*y^2+y^3
radsimp:
sin(x)^2+sin(y)^2+x^3+3*x^2*y+3*x*y^2+y^3
combine(trig):
1-1/2*cos(2*x)-1/2*cos(2*y)+x^3+3*x^2*y+3*x*y^2+y^3
factor:
sin(x)^2+sin(y)^2+x^3+3*x^2*y+3*x*y^2+y^3
expand:
sin(x)^2+sin(y)^2+x^3+3*x^2*y+3*x*y^2+y^3
combine:
1-1/2*cos(2*x)-1/2*cos(2*y)+x^3+3*x^2*y+3*x*y^2+y^3
convert(exp):
-1/4*(exp(i*x)-1/exp(i*x))^2-1/4*(exp(i*y)-1/exp(i*y))^2+x^3+3*x^2*y+3*x*y^2+y
^3
convert(sincos):
sin(x)^2+sin(y)^2+x^3+3*x^2*y+3*x*y^2+y^3
convert(tan):
4*tan(1/2*x)^2/(1+tan(1/2*x)^2)^2+4*tan(1/2*y)^2/(1+tan(1/2*y)^2)^2+x^3+3*x^2*
y+3*x*y^2+y^3
collect(x):
sin(x)^2+sin(y)^2+x^3+3*x^2*y+3*x*y^2+y^3
mwcos2sin:
sin(x)^2+sin(y)^2+x^3+3*x^2*y+3*x*y^2+y^3
ans =
sin(x)^2+sin(y)^2+x^3+3*x^2*y+3*x*y^2+y^3
```



函数 simplify 使用一般的化简方法进行化简计算，函数 factor 的作用是把表达式化成连乘积的形式，函数 expand 用来把表达式展开，函数 collect 用来进行降幂排列。convert(exp)，convert(sincos)和 convert(tan)分别以 exp，sin&cos 和 tan 三种数学函数展开，mwcos2sin 表示给出一个利用 cos 与 sin 函数组成的化简结果，

radsimp 表示尽量给出一个复数形式的结果，combine(trig)和 combine 表示利用三角函数形式把高次三角函数转化为一次三角函数组合的形式。

4.4.1.7 表达式展开为重叠形式

函数 horner 可以把表达式展开为重叠形式，如 ax(bx(cx-d)+e)+g，调用格式为：

```
horner(P)
```

参数说明： P 是一个符号表达式，比如执行 horner(sym('x^3-6*x^2+11*x-6'))将得到-6+(11+(-6+x)*x)*x。

4.4.1.8 把符号表达式化简为有理分式

函数 numden 可以把符号表达式化简为有理分式的形式，其调用格式为：

```
[n, d] = numden(A);
```

参数说明： A 是符号表达式。输出 n 为分子的表达式，d 为分母的表达式。比如[n,d] = numden(sym('x/y + y/x'))，将得到 n=x^2+y^2 和 d=x*y。

4.4.1.9 相同因子的筛选与代换

函数 subexpr 可以用来完成表达式中相同因子的筛选并进行代换以得到简短的表达式，其调用格式为：

```
[r,s] = subexpr(t,'s');
```

参数说明： 这里 t 是表达式。s 用来代换相同因子的名称以及在代换后表达式中相同因子的名称。r 是代换后的表达式。

例 4-17： 比较 solve 和 subexpr 函数的作用。

```
t = solve('a*x^3+b*x^2+c*x+d = 0');
[r,s] = subexpr(t,'s');
```

用户可以查看代换前表达式 t 和代换后表达式 r 以及代换因子 s。

4.4.2 赋值函数的使用

下面介绍几个赋值函数的用法。我们在 MATLAB 编程过程中用得比较多的赋值函数包括 subs，ccode，fortran 等，下面分别进行讲解。

4.4.2.1 变量赋值

函数 subs 可以用来实现对符号表达式中变量进行赋值的操作，其调用格式为：

```
v = subs(s,old,new);
v = subs(s,{var1,var2,var3,…},{v1,v2,v3,…});
```

参数说明： v 是赋值后的表达式。s 是表达式。old 是要代换的变量名。new 是更换后的值。var1，var2，var3 等是多个符号变量名，v1，v2，v3 等是相应的值。

例 4–18：下面是 subs 函数应用的两个例子。

```
syms a b c;
f1 = subs(a+b,a,3);
f2 = subs(sin(a)+cos(b),{a,b},{1,c});
```

上述程序输出：

```
f1 = 3+b
f2 =sin(1)+cos(c)
```

4.4.2.2 转化 C 源代码

函数 ccode 可以把符号表达式转化为对应的 C 语言源代码表达式，调用格式如下：

```
ccode(f);
```

参数说明：f 是一个符号表达式。

例 4–19：下面是一个调用函数 ccode 的例子。

```
f = taylor(sym('log(1+x)'),4);          % 进行泰勒级数展开
ccode(f);                               % 把 f 转化为 C 代码
```

上述程序输出如下：

```
ans =      t0 = x-x*x/2.0+x*x*x/3.0;
```

4.4.2.3 转化为 fortran 源代码

函数 fortran 可以把符号表达式转化为对应的 fortran 语言源代码表达式，调用格式为：

```
fortran(g)
```

参数说明：f 是一个符号表达式。

例 4–20：调用函数 fortran。

```
g = diff(sym('sin(x^3)'));          % 对表达式进行微分计算
fortran(g);                         %把符号表达式转化为 fortran 代码
```

上述程序输出如下：

```
ans =      t0 = 3*cos(x**3)*x**2
```

4.4.3 符号微积分

在 MATLAB 中利用 diff,jacobian,int 和 dsolve 等函数可以进行微分和积分运算。此外利用 limit 函数还可以进行极限的计算。下面将具体介绍这些函数的用法。

4.4.3.1 计算微分

函数 diff 可以用来计算符号表达式的微分，其调用格式如下：

```
df = diff(f,n);
```

参数说明：df 是微分计算的结果。f 是输入的表达式。n 表示求导的次数，其默认值为 1。

例 4-21：调用 diff 函数。

```
df = diff(sym('x*log(x+1)'),2)
```

上述语句输出如下：

```
df =2/(1+x)-x/(1+x)^2
```

4.4.3.2 计算雅可比矩阵

函数 jacobian 计算符号表达式的雅可比矩阵，对于由变量 v_1，v_2，…，v_n 组成的函数向量[f_1，f_2，…，f_m]而言，它的雅可比矩阵 J 的各个元素可以按如下公式计算。

$$J_{m,n}=\frac{\partial f_m}{\partial v_n}$$

在 MATLAB 中函数 jacobian 的调用格式如下：

```
Jf = jacobian(f,v)
```

参数说明：f 是符号表达式。v 是由不同变量名组成的向量。当 v 中只含有一个变量时，jacobian(f,v)和 diff(f)是等价的。

例 4-22：调用函数 jacobian。

```
Jf1 = jacobian(sym('x*log(1+x)'),sym('x'));
Jf2 = jacobian(sym('[x+y,x*y]'),[sym('x'),sym('y')]);
```

上述语句输出如下：

```
Jf1 =log(1+x)+x/(1+x)
Jf2 = [ 1, 1]
[ y, x]
```

这里用户也可以利用 syms 函数来定义符号变量。

4.4.3.3 计算不定积分和定积分

函数 int 可以用来计算不定积分和定积分，该函数调用格式如下：

```
v = int(S);
v = int(S,var);
v = int(S,a,b);
v = int(S,var,a,b);
```

参数说明：v 是返回的积分结果。S 是被积函数，可以是一个向量或者矩阵。var 是对积分表达式中的符号变量名进行积分。a 和 b 用于指定积分的上下限。当 var 缺省时，函数 int 将默认对 x 进行积分。

例 4-23：下面是 int 函数的 4 个例子。

```
syms x y a
v1 = int([sin(x),x*exp(x)])
v2 = int(y*sin(x*y),y)
v3 = int([sin(x)/x,x*exp(-x^2)],0,inf)
v4 = int(sin(x*a)/x,x,0,1)
```

上述程序计算后返回如下结果：

```
v1 =[ -cos(x),  (-1+x)*exp(x)]
v2 =1/x^2*(sin(x*y)-y*cos(x*y)*x)
v3 =[ 1/2*pi,    1/2]
v4 =sinint(a)
```

结果中的函数 sinint 定义为 sinint(x) = int(sin(t)/t,t,0,x)。在函数 int 计算结果中可能会出现一些特殊函数，用户可以使用 help 或者 mhelp 查看特殊函数的意义。函数 int 得到的结果是符号型数据，如果用户希望输出的是 double 型数据，可以利用函数 double 来转化。

4.4.3.4 求解微分方程

函数 dsolve 直接的用法是求解微分方程。这里介绍一下如何利用它来计算定积分。对于一个定积分有：

$$\int_a^b f(x)=F(b)-F(a) \quad (F'(x)=f(x))$$

通过函数 dsolve 可以计算出被积函数 $f(x)$ 的原函数 $F(x)$，然后再带入积分上限和下限进行简单计算就可以得到积分结果了。

例 4-24： 计算定积分。

$$\int_1^2 \sin t \cos t dt$$

相关程序如下：

```
F=dsolve('Df = sin(t)*cos(t)');
v=subs(F,2)-subs(F,1)
```

输出结果为：

```
v =-1/4*cos(4)+1/4*cos(2)
```

这里 v 是一个符号变量，利用 double 函数可以把这个表达式转化为 double 数据。

4.4.3.5 计算极限

函数 limit 可以用来计算不同类型的极限，其调用格式为：

```
v = limit(f);
v = limit(f,a);
v = limit(f,x,a)
v=limit(f,x,a, 'right');
v=limit(f,x,a, 'left');
```

参数说明： v 是返回的极限值。f 是符号表达式，它可以是单个表达式，也可以是由多个表达式组成的向量或者矩阵。x 指定符号表达式中 x 为变量。a 用于指定求 x 趋于 *a* 的极限值。right 和 left 分别用于指定求表达式的右极限和左极限。

例 4-25： 利用函数 limit 进行极限计算。

```
syms x a t;
v1 = limit(sin(x)/x)                      % 计算表达式在 x 趋于 0 时的极限
```

```
v2 = limit((x-1)/(x^2-1),1)                 % 计算表达式在 x 趋于 1 时的极限
v3 = limit((1+2*x/t)^(3*t),t,inf)           % 计算表达式在 t 趋于无穷大 ( inf ) 时的极限
v4 = limit(1/x,x,0,'right')                 % 计算表达式在 x 趋于 0 时的右极限
v5 = limit([1/x,abs(x)/sin(x)],x,0,'left')  % 计算表达式在 x 趋于 0 时的左极限
```

输出结果为：

```
v1 =1
v2 =1/2
v3 =exp(6*x)
v4 =Inf
v5 =[ -Inf,   -1]
```



上面的例子表明函数 limit 可以计算数学表达式的极限，同时还可以计算左极限和右极限。此外输入函数的形式还可以是向量或者矩阵。

4.5 多项式运算

在数学上，多项式是一类基本的数学函数，因为它简单且可组成完备函数基，因此在很多研究中用它来作为复杂函数的近似形式。在 MATLAB 中提供了多种关于多项式操作的函数，比如 roots，poly，polyval，residue，polyfit，polyder 和 conv 等，用户利用这些函数可以便捷地对多项式进行操作。本节详细说明如何利用这些函数进行与多项式相关的计算。

4.5.1 多项式的定义

对于一个多项式的降幂排序表达式：

$$P_n(x)=a_nx^n+a_{n-1}x^{n-1}+...+a_1x+a_0 \tag{4-1}$$

在 MATLAB 中把系数 a_n，a_{n-1}，…，a_1，a_0 提取出来，按次序对应一个向量，即：

$$p=[a_n,a_{n-1},...,a_1,a_0] \tag{4-2}$$

这样可以在向量和多项式系数之间建立起联系，缺少的幂次对应的系数认为是 0。比如，$p_4(x)=x^4+x^2-4$ 可以把系数向量写为 p=[1,0,1,0,-4]，其中立方项和一次项是缺少的。对于向量，在 MATLAB 中完全可以进行灵活的处理了。

4.5.2 特殊函数与特殊多项式

本节介绍一些特殊函数和特殊多项式的计算，它们在一些问题研究中往往由于其特殊性质而可以作为函数基，从而使求解过程变得简单。

4.5.2.1 厄米多项式

厄米多项式是一类特殊的多项式，不同阶次的多项式之间满足正交关系，其定义公式（母函数）为：

$$H_n(x)=(-1)^n e^{x^2}\frac{\mathrm{d}^n}{\mathrm{d}x^n}e^{-x^2} \tag{4-3}$$

其中，n 是自然数。

前 5 个厄米多项式的表达式为：

$H_1(x)=1$

$H_2(x)=4x^2-2$

$H_3(x)=8x^3-12x$

$H_4(x)=16x^4-48x^2+12$

$H_5(x)=32x^5-160x^3+120x$

可见奇数阶（除 1 阶外）厄米多项式是奇函数，偶数阶厄米多项式是偶函数。

关于厄米多项式计算的 MATLAB 程序下载的网址读者可以在附录 A 中查询。

4.5.2.2 勒让德多项式

勒让德多项式的定义为：

$$L_n(x)=\frac{1}{2^n n!}\frac{\mathrm{d}^n}{\mathrm{d}x^n}\left(x^2-1\right)^n \tag{4-4}$$

前 5 个勒让德多项式的表达式为：

$L_0(x)=1$

$L_1(x)=x$

$L_2(x)=\left(3x^2-1\right)/2$

$L_3(x)=\left(5x^3-3x\right)/2$

$L_4(x)=\left(35x^4-30x^2+3\right)/8$

勒让德多项式定义于[-1,1]，同时 $L_n(1)=1$。

关于勒让德多项式计算的 MATLAB 程序下载的网址读者可以在附录 A 中查询。

4.5.2.3 格根包尔（Gegenbauer）多项式

格根包尔（Gegenbauer）多项式如下定义：

$$C_n^{(\alpha)}(x)=\frac{(2\alpha)^n}{n!}\,{}_2F_1\left(-n,2\alpha+n,\alpha+\frac{1}{2},\frac{1-x}{2}\right) \tag{4-5}$$

其中，$(2\alpha)^n=\Gamma(2\alpha+1)/\Gamma(2\alpha-n+1)$，$\Gamma(\ldots)$表示伽马（gamma）函数，在 MATLAB 中伽马函数的计算可以用函数 gamma。${}_2F_1(\ldots)$是合流超几何函数，在 MATLAB 中可以利用 hypergeom 函数来计算。

4.5.2.4 雅克比（Jacobi）多项式

雅克比（Jacobi）多项式如下定义：

$$J_n^{(\alpha,\beta)}(x)=\frac{\Gamma(\alpha+n+1)}{n!\Gamma(\alpha+\beta+n+1)}\sum_{m=0}^{n}\frac{\Gamma(x+1)}{\Gamma(n+1)\Gamma(x-n+1)}\frac{\Gamma(\alpha+\beta+n+m+1)}{\Gamma(\alpha+m+1)}\left(\frac{x-1}{2}\right)^m \tag{4-6}$$

因此，雅可比多项式利用 gamma 函数进行求和就可以算出来。

4.5.2.5 拉盖尔多项式

拉盖尔多项式定义为：

$$L_n(x)=\frac{e^x}{n!}\frac{d^n}{dx^n}x^n e^{-x}=\sum_{m=0}^{n}\frac{\Gamma(n+1)}{\Gamma(m+1)\Gamma(n-m+1)}\frac{(-x)^m}{m!} \tag{4-7}$$

利用 LaguerrePoly.m 函数来计算拉盖尔多项式系数的程序的下载网址读者可以在附录 A 中查询到。

4.5.2.6 特殊函数的函数值计算

在 MATLAB 中提供了 mfun 函数来计算一些特殊函数的函数值，该函数调用格式为：

```
y = mfun('fun',p1,p2,...,pk);
```

参数说明： y 是返回的函数值。fun 是特殊函数名称的缩写。p1，p2，…，pk 是特殊函数的参数，变量的个数因函数而异，用户使用的时候需要注意。下面列出 mfun 可以计算的一些特殊函数及用法，如表 4.4 所示。

表 4.4 mfun 函数使用说明

调用格式	说明
mfun('bernoulli',n);	贝努力数
mfun('bernoulli',n,z);	贝努力多项式
mfun('BesselI',x1,x);	第一类修正的贝赛尔函数
mfun('BesselJ',x1,x);	第一类贝赛尔函数
mfun('BesselK',x1,x);	第二类修正的贝赛尔函数
mfun('BesselY',x1,x);	第二类贝赛尔函数
mfun('Beta',z1,z2);	Beta 函数
mfun('binomial',x1,x2);	二项式系数
mfun('EllipticF',z,k);	第一类不完全椭圆积分
mfun('EllipticK',k);	第一类完全椭圆积分
mfun('EllipticCK',k);	第一类互余完全椭圆积分
mfun('EllipticE',k);	第二类完全椭圆积分
mfun('EllipticE',z,k);	第二类不完全椭圆积分
mfun('EllipticCE',k);	第二类互余完全椭圆积分
mfun('EllipticPi',nu,k);	第三类完全椭圆积分
mfun('EllipticPi',z,nu,k);	第三类不完全椭圆积分
mfun('EllipticCPi',nu,k);	第三类互余完全椭圆积分
mfun('erfc',z);	余误差函数
mfun('erfc',n,z);	余误差函数的迭代积分
mfun('Ci',z);	余弦积分
mfun('dawson',x);	道森（dawson）积分
mfun('Psi',z);	双函数
mfun('dilog',x);	二重对数积分
mfun('erf',z);	误差函数
mfun('euler',n);	欧拉数
mfun('euler',n,z);	欧拉多项式
mfun('Ei',x);	指数积分
mfun('Ei',n,z);	指数积分
mfun('FresnelC',x);	菲涅尔余弦积分
mfun('FresnelS',x);	菲涅尔正弦积分

（续表）

调用格式	说明
mfun('GAMMA',z);	Gamma 函数
mfun('harmonic',n);	谐函数
mfun('Chi',z);	双曲余弦积分
mfun('Shi',z);	双曲正弦积分
mfun('GAMMA',z1,z2);	不完全 Gamma 函数
mfun('LambertW',z);	朗伯函数
mfun('LambertW',n,z);	朗伯函数
mfun('lnGAMMA',z);	Gamma 函数的对数
mfun('Li',x);	对数积分
mfun('Psi',n,z);	多函数
mfun('Ssi',z);	移位正弦积分
mfun('Si',z);	正弦积分
mfun('Zeta',z);	黎曼 Zeta 积分
mfun('Zeta',n,z);	黎曼 Zeta 积分
mfun('Zeta',n,z,x);	黎曼 Zeta 积分



上述特殊函数使用帮助的查看是通过 mhelp functionname 进行的，functionname 是上表中引号部分的内容。

4.5.2.7 其他多项式和函数

对于切比雪夫多项式、格根包尔多项式、厄米多项式、雅可比多项式、拉盖尔多项式和勒让德多项式等函数可以通过下面的形式调用：

```
maple('T',n,x);              % 第一类切比雪夫多项式
maple('U',n,x);              % 第二类切比雪夫多项式
maple('G',n,x1,x);           % 格根包尔多项式
maple('H',n,x);              % 厄米多项式
maple('P',n,x1,x2,x);        % 雅可比多项式
maple('L',n,x);              % 拉盖尔多项式
maple('L',n,x1,x);           % 广义拉盖尔多项式
maple('P',n,x);              % 勒让德多项式
```

参数说明： n，x1，x2，x 是这些多项式函数的参数，用户可以通过 mhelp functionname 来查看具体的使用信息。需要注意的是，在调用上面语句之前需要使用语句 maple('with','orthopoly') 来加载这些函数，比如下面的例子。

例 4–26： 多项式和函数的应用。

```
maple('with','orthopoly')    % 加载特殊函数
v=maple('H',1,2)    % 利用函数 maple 计算厄米多项式
```

上述语句输出如下：

```
ans =[G, H, L, P, T, U]
v =4
```



ans给出所加载函数的函数名称简写形式，v表示相应特殊函数的函数值。

4.5.3 多项式的运算

本小节主要是介绍关于多项式计算的一些函数，比如求根、求多项式函数值、卷积、求导和分式展开等。

4.5.3.1 零点计算

利用函数roots可以方便地计算多项式函数的零点，该函数调用格式如下：

```
r = roots(p);
```

参数说明： r是返回的多项式函数零点，为一个列向量。p是目标多项式的系数向量，用户可以定义向量p为行向量（也可以是列向量）。这里补充一个函数poly，它可以根据多项式的零点得到多项式的系数向量。在过程上看，函数roots和poly是一对互逆函数。此外，利用函数poly还可以得到方矩阵的特征多项式的系数向量。

例4–27： 多项式函数的零点。

```
A=roots([1,0,0,0,3])
```

输出为：

```
A =  -0.9306 + 0.9306i
    -0.9306 - 0.9306i
  0.9306 + 0.9306i
    0.9306 - 0.9306i
```



这个得到的结果对应着"–3"的4次方根。读者可以用"A.^4"来验证这个关系。

4.5.3.2 标准字符串表达式

函数poly2str可以根据一个多项式系数向量得到多项式的标准字符串表达式，其调用格式为：

```
sx = poly2str(p,'s');
[sx,len] = poly2str(p,'s');
```

参数说明： sx是多项式的字符串形式。p是多项式系数形式。s用于制定自变量的名称，可以是其他英文字母或者多个字母。len表示字符串sx的长度，即len = length(sx)。但是利用poly2str得到的字符串形式不是一个合法的MATLAB表达式，例如：

```
>> sx = poly2str(1:4,'x')
sx =   x^3 + 2 x^2 + 3 x + 4
```

可以通过下面的语句来把sx转化为合法的字符串。

```
sx = poly2str(1:4,'x');
```

```
sx = fliplr(deblank(fliplr(sx)));          % 删去 sx 前面的空格符
sxx = strrep(sx,' x','*x')                  % 补上乘号
```

上述程序执行后得到的结果如下：

```
sxx =2*x^3 + 3*x^2 + 4*x + 5
```

这样可以利用 eval 或者 inline 函数来对字符串 sxx 做进一步处理。

4.5.3.3 特殊点的函数值计算

函数 polyval 计算多项式函数在某些特殊点的函数值，该函数的调用格式如下：

```
y = polyval(p, x);
```

参数说明：y 是返回的函数值。p 是多项式的系数向量。x 是自变量的离散采样点，x 可以是向量。

例 4-28：特殊点的函数值计算。

```
p=[1,2,3];
x=0:.2:1;
y=polyval(p,x)
```

输出为：

```
y =    3.0000    3.4400    3.9600    4.5600    5.2400    6.0000
```



这个例子用来计算多项式 x^2+2x+3 在指定位置 x 处的函数值。

4.5.3.4 卷积和解卷积计算

多项式的乘法和除法运算就对应于卷积和解卷积计算，MATLAB 提供了函数 conv 和 deconv 来分别进行卷积和解卷积运算。函数 conv 的调用格式如下：

```
c = conv(p1,p2);
```

参数说明：c 是卷积计算得到的多项式的系数向量。p1 和 p2 是两个多项式的系数向量，要求 p1 和 p2 是浮点型数据。3 个系数向量的长度关系是 c = length(p1)+length(p2)-1。

对于多项式除法（解卷积运算），比如

$$\frac{p_1(x)}{p_2(x)}=q(x)+\frac{r(x)}{p_2(x)} \tag{4-8}$$

实现上面的计算可以利用函数 deconv，其语法格式为：

```
[q, r] = deconv(p1, p2);
```

参数说明：q 对应于多项式 $q(x)$的系数向量。r 是余项多项式 $r(x)$的系数。p1 和 p2 分别是多项式 $p_1(x)$和 $p_2(x)$的系数向量。

例 4-29：解卷积运算。

```
p1=[1,6,12,10];
```

```
p2=[1,4,4];
[q,r]=deconv(p1,p2)
```

输出为：

```
q =     1     2
r =     0     0     0     2
```



这个例子用来计算多项式 $x^3+6x^2+12x+10$ 与多项式 x^2+4x+4。通过简单的因式分解可以知道商式为 $x+2$，余式为 2，而上述结果正好与这个对应。

4.5.3.5 导数计算

多项式 $P_n(x)=a_nx^n+a_{n-1}x^{n-1}+...+a_1x+a_0$ 的一次导数可以写为：

$$\frac{dP_n(x)}{dx}=na_nx^{n-1}+(n-1)a_{n-1}x^{n-2}+...+a_1$$

在 MATLAB 中可以利用函数 polyder 来计算多项式函数的导数，其调用格式为

```
dp = polyder(p);                    %格式 1
dp = polyder(p1, p2);               %格式 2
[dq, dr] = polyder(p1, p2);         %格式 3
```

参数说明：dp 是导数多项式的系数向量。p 是多项式的系数向量。根据上面的推导可以得出 dp 和 p 之间的关系是 dp=p(1:end-1).*(length(p)-1:-1:1)。格式 2 用来计算两个多项式乘积的导数。格式 3 用来计算两个多项式相除的导数，其中 dq 是分子多项式的系数向量，dr 是分母多项式的系数向量。

例 4–30：导数计算。

```
p1=[1,2,3];
p2=[1,4];
d1=polyder(p1)
d2=polyder(p1,p2)
```

输出为：

```
d1 =     2     2
d2 =     3    12    11
```



p1 对应于多项式 x^2+2x+3 的系数，p2 对应于多项式 $x+4$ 的系数，而 d1 对应于多项式 p1 的导数，d2 是两个多项式乘积的导数。

4.5.3.6 公式转换

函数 residue 可以把分式多项式按如下方式展开：

$$\frac{B(x)}{A(x)}=\frac{r_1}{x-p_1}+\frac{r_2}{x-p_2}+...+\frac{r_n}{x-p_n}+k(x) \tag{4-9}$$

利用函数 residue 实现上述公式转换的用法如下：

```
[r, p, k] = residue(B, A);
```

此外，函数 residue 还可以实现上述公式相反方向的转化，如：

```
[B, A] = residue(r, p, k);
```

即实现了部分最简化分式形式向分式多项式转化的过程，函数 residue 根据输入、输出参数格式的差异进行不同过程的转化计算。

例 4-31： 公式转换应用实例。

```
A=[1,-5,6];
B=[1,5,6];
[r,p,k]=residue(A,B)
```

输出为：

```
r =  -30.0000
   20.0000
p =   -3.0000
  -2.0000
k =    1
```



向量 A 对应着多项式红 x^2-5x+6 的系数，B 对应着多项式 x^2+5x+6 的系数，读者可以根据输出验证 residue 函数的运算关系。

4.6 卷积与相关

在 4.5.3 小节介绍了计算多项式卷积的函数 conv，本节主要介绍利用快速傅里叶变换来计算卷积和相关计算。卷积和相关运算在一些信号处理问题中具有重要作用，其中卷积可以描述一些系统输入和输出之间的关系，而相关运算可以应用在目标识别中。

二维卷积运算可以按下面的公式转化为利用傅里叶变换计算：

$$C(x,y)=f(x,y)\otimes g(x,y)=\iint_{u,v} f(x-u,y-v)g(u,v)\mathrm{d}u\mathrm{d}v=\mathrm{F}^{-1}\left[\mathrm{F}\{g(x,y)\}\mathrm{F}\{f(x,y)\}\right] \tag{4-10}$$

符号⊗表示卷积。F^{-1} 和 F 分别代表逆傅里叶变换和傅里叶变换，因此可以利用快速傅里叶变换来计算两个函数的卷积，而这种卷积也被称为循环卷积。

二维的相关运算可以表示为如下形式：

$$K(x,y)=f(x,y)\oplus g(x,y)=\iint_{u,v} f(u+x,v+y)g(u,v)\mathrm{d}u\mathrm{d}v=\mathrm{F}^{-1}\left[\mathrm{F}\{g(x,y)\}\mathrm{F}^{*}\{f(x,y)\}\right] \tag{4-11}$$

符号⊕表示相关运算。符号*表示取复共轭。可以看出利用快速傅里叶变换能够计算两个函数的卷积。

4.6.1 计算二维离散卷积

利用函数 conv2 可以计算二维离散卷积，其调用格式为：

```
C = conv2(A,B);
C = conv2(A,B,shape);
```

参数说明： C是返回的相关系数矩阵。A和B是输入的两个矩阵， A的行数和列数分别是m1和n1，B的行数和列数分别是m2和n2。shape是字符串，用于限制输出矩阵的大小，其取值为：full为默认值，此时返回矩阵的行数和列数分别为m1+m2-1和n1+n2-1；same，返回一个与矩阵A大小相同完整卷积矩阵的中间部分；valid，当矩阵A的行数和列数都不小于矩阵B的行数和列数时，矩阵C的行数和列数为m1-m2+1和n1-n2+1，否则C= []。

例4-32： 计算二维离散卷积。

```
A=magic(3);
B=[1,1,1;2,2,2;3,3,3];
C=conv2(A,B,'same')
```

输出为：

```
C =   26    45    26
      56    90    56
      50    75    58
```



读者可以根据卷积的定义验证上面的计算结果，这里使用参数same是为了得到和输入矩阵大小一样的矩阵。

4.6.2 计算线相关系数

函数corr可以计算线相关系数，其调用格式为：

```
R = corr(A,B);
```

参数说明： R是返回的线相关系数矩阵。A和B是两个输入矩阵。

例4-33： 计算两个函数的卷积。

```
[x,y] = meshgrid(linspace(-3,3,40));    % 生成坐标矩阵
z1 = exp(-[x.^2+y.^2]);                  % 得到第1个函数的值
z2=peaks(40);                            % 利用peaks函数生成第2个函数
C1 = ifft2(fft2(z1).*fft2(z2));          % 计算循环卷积
C2 = ifft2(fft2(z1).*conj(fft2(z2)));    % 计算相关
c1 = conv2(z1,z2,'same');                % 二维离散卷积
c2 = corr(z1,z2);                        % 线相关运算
figure;
subplot(221);                            % 打开第1个坐标轴
mesh(abs(C1));                           % 绘出C1的模值曲面
colormap([0,0,0]);                       % 设置为黑色网格曲线
xlabel('(a)','Fontsize',16);             % 坐标轴标注
subplot(222);                            % 打开第2个坐标轴
mesh(abs(C2));                           % 绘出C2的模值曲面
xlabel('(b)','Fontsize',16);             % 坐标轴标注
set(gcf,'Color','w')                     % 设置图形窗口背景为白色
subplot(223);                            % 打开第3个坐标轴
mesh(c1);                                % 绘出c1的模值曲面
xlabel('(c)','Fontsize',16);             % 坐标轴标注
```

```
subplot(224);                        % 打开第 4 个坐标轴
mesh(c2);                            % 绘出 c2 的模值曲面
xlabel('(d)','Fontsize',16);         % 坐标轴标注
```

得到的图形如图 4.2 所示，从中读者可以比较相关与卷积结果的差异性。同时，利用傅里叶变换计算的卷积和相关结果与 MATLAB 提供的离散形式的计算函数存在一定差异，在使用时需要考虑到这一点。

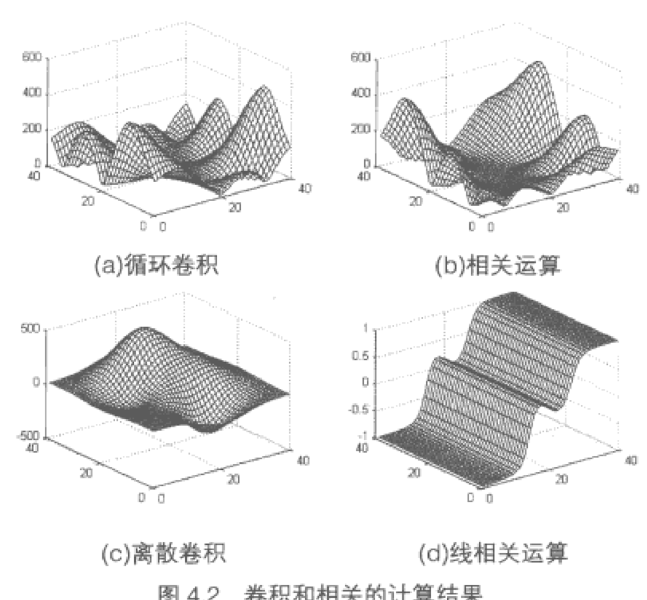


图 4.2　卷积和相关的计算结果

4.7　表达式的应用技巧

本节介绍在不同类型表达式之间转化的技巧，这些对于灵活地解决一些问题很有帮助。

4.7.1　符号表达式转化为字符串

在 MATLAB 中，可以利用 char 函数实现把符号表达式转化为字符串。

例 4-34：把符号表达式转化为字符串。

```
f=dsolve('Df = f + sin(t)', 'f(pi/2) = 0');
f=char(f);
```

此时，f 是一个字符串型数据，用户可以把它转化为 inline 函数，或者用函数 favel 和 eval 进行求值等操作。

4.7.2　对变量的调用

在编程的时候可以通过交换函数 subs 来调用变量名和对应的值。

例 4-35：subs 函数实现变量名和对应值的转变。

```
f1=subs(sym('a*u+b'),{'a','b'},{'3','2'})
```

```
f2=subs(sym('a*u+b'),{'3','2'},{'a','b'})
```

输出结果为：

```
f1 =3*u+2
f2 =3*u+2
```

4.7.3 含变化参数的符号计算

在进行符号计算的过程中，当要求某参数不断变化时，可以利用 num2str 函数来实现，比如：

```
a=2;
v = int(sym(['x^',num2str(a),'+3']),1,2)
```

输出结果为：

```
v = 16/3
```

这样我们就可以不断地变化 a 的数值了。

4.7.4 用函数实现赋值

在 MATLAB 中可以利用 eval 函数完成赋值功能。

例 4-36：利用 eval 函数完成赋值。

```
syms x y
f = x^2+x+2
df = diff(sym('x^3+y^2*x'));
x = 1;
y = 2;
fv = eval(df)
```

输出结果为：

```
fv =     7
```

4.7.5 调用 maple 函数来计算

在 MATLAB 中内置 mfun 函数，利用它可以实现对 maple 函数的调用，以完成相关计算。

例 4-37：maple 函数的应用。

```
str2num(maple('evalf(Psi(1.2))'))
```

这是一个计算双函数语句，用户可以更换函数名去计算其他特殊函数。这里我们可以像使用 maple 软件一样来计算一些函数值。

4.7.6 符号表达式的转化

在 MATLAB 中可以将符号表达式转化为 inline 函数。

例 4-38：转化为 inline 函数。

```
syms x y
```

```
f = x^2+y*3;
fun = inline(char(f))
```

输出如下：

```
fun =     Inline function:
      fun(x,y) = x^2+3*y
```

4.7.7 数值型矩阵转化为符号矩阵

利用函数 sym 可以把数值型矩阵转化为符号矩阵，比如：

```
>>symb=sym(magic(3))
```

输出如下：

```
symb =
[ 8, 1, 6]
[ 3, 5, 7]
[ 4, 9, 2]
```

4.7.8 复合函数的应用

例 4-39：利用函数 compose 和 subs 进行复合函数的推导。

```
syms x y;
f = 1/(1 + x^2); g = sin(y);
fa = compose(f,g)
fb = subs(f,x,g)
```

输出如下：

```
fa = 1/(1+sin(y)^2)
fb =1/(1+sin(y)^2)
```

可见利用这两个函数都可以进行复合函数的计算，其中函数 compose 是专门用于计算复合函数的。

4.7.9 建立抽象函数

利用函数 sym 和 maple 可以建立抽象函数，下面是两个例子：

```
f = sym('f(x,y)')
g = maple('map(g,x,y)')
```

输出如下：

```
f =f(x,y)
g =g(x,y)
```

这里 f 和 g 的类型是符号型，它们没有具体的函数关系式，只是一个抽象的函数形式，因此叫做抽象函数。用户可以进一步对抽象函数进行符号计算。

此外，MATLAB 提供了函数 funtool 供用户考察两个一元函数的性质及它们之间的关系。利用

这个界面用户可以观测到设定区间内的函数曲线，同时能够进行一些简单的函数运算。

4.8 小结

本章主要介绍了不同类型表达式的建立和特殊函数的相关知识，详细介绍了算术表达式、关系表达式和逻辑表达式的知识，熟悉这些表达式对于编写程序非常重要。接下来还介绍了进行符号计算的知识，首先是符号变量的建立，然后是符号运算的相关函数及用法。特殊函数，包括多项式函数，如何利用 MATLAB 计算这些特殊函数，以及卷积和相关这两种运算都在本章有所涉及。

◆ ◆ ◆

第 5 章　程序结构与优化

本章包括

- ◆ **条件语句**　具体包括条件语句的建立和等效改进方法。
- ◆ **switch 语句**　介绍该语句的用法及主要需要注意的地方。
- ◆ **循环结构**　描述两种循环结构，即 for 循环和 while 循环。
- ◆ **递归结构**　通过两个例子介绍递归结构的设计与调试。
- ◆ **人机交互函数**　介绍几种用户和计算机互动操作的函数。
- ◆ **程序加速**　概括加速 MATLAB 程序的几个编程技巧。
- ◆ **程序注释**　描述程序注释需要注意的事项及如何更好地利用这一功能。
- ◆ **常见错误的调试**　讲述调试程序需要注意的一些事项。

在大型程序编写过程中，一个清晰的程序结构可以帮助用户快速地建立程序。MATLAB 程序总体可以分为顺序、循环、分支和递归 4 种结构，这和其他高级语言是相似的。顺序结构是依次按先后顺序执行程序的每一条语句，其中并列语句的先后顺序可以交换。语句在程序文件中的次序就反映了程序的执行顺序。典型的顺序结构是不含有其他结构（如循环和分支结构）的批处理文件或是 MATLAB 中的命令文件。虽然大多数程序都包含许多子结构，但从整体上看，它们大多属于顺序结构。在这个结构中，计算任务依次对应显示各条语句，即依次执行各条语句。循环和分支结构将在下面几节中详细介绍。一般情况，MATLAB 程序由这 4 种结构就可以建立起来。MATLAB 的程序结构与 C 语言大体相似，而且比 C 语言的编程风格要简化很多，编译过程更人性化，随着版本的升级，界面的功能也不断增加和完善。在编程开始时，用户最好在稿纸上先勾画出程序的大体结构，熟练的用户也需要在大脑中构思出整体轮廓，根据选择最优结构、避免重复计算的原则来完成程序的编写工作。本章将详细介绍 MATLAB 中程序结构方面的知识，辅助读者写出简洁的 MATLAB 程序。

5.1　条件语句

条件语句用于因满足不同条件而执行不同操作的情况，是一种分支结构。在 MATLAB 中利用 if...end 结构来实现条件语句。其基本调用格式如下：

```
if expression1
   statements
elseif expression2
   statements
else
   statements
end
```



其中 expression1 和 expression2 可以是变量或者表达式形式。满足条件时将执行相应的语句 statements。特别地，表达式的值是非零值时将认为条件成立，执行相应的语句。当判断条件为逻辑表达式的时候，需要注意左右相等的判断，如"A==1"，当 A 是一个 double 型数据的时候，这个判断可能会导致错误。如果 A 预期为 0 和 1 两个可能值，那么可以使用"A>0.5"来替换"A==1"。当需要多个逻辑表达式组合的时候，可以分别用"&"和"|"来表示"与"和"或"的关系。逻辑操作"非"利用符号"~"来实现。超过两个逻辑表达式的时候需要使用方括号或者圆括号来设定计算顺序。其中，elseif 和 else 部分是可以缺省的，读者根据实际情况确定条件语句的使用格式。而且当存在多个判断情况时，可以增加 elseif 的个数来建立条件语句。

例 5-1： 利用条件语句计算下面二元函数的函数值。

$$f(x,y)=\begin{cases}0, & x=0\text{或}y=0\\ x^2y, & x<0\text{且}y<0\\ xy^2, & x<0\text{且}y>0\\ x^2y^2, & x>0\text{且}y<0\\ x^2y^3, & x>0\text{且}y>0\end{cases}$$

分析： 该二元函数共分为 5 种情况，利用 if ... elseif ... else... end 结构来实现这个二元函数的定义。其中每一种情况的判断条件需要用到两个逻辑表达式的组合，因此这个例子的实现需要一个完整的条件语句结构。相应程序如下：

```
% x, y 的定义略去
if x==0 | y==0;
   f = 0;
elseif x<0 & y<0;          % 判断第三象限
   f = x^2*y;              % 赋值
elseif x<0 & y>0;          % 判断第二象限
   f = x*y^2;              % 赋值
elseif x>0 & y<0;          % 判断第四象限
   f = x^2*y^2;            % 赋值
else                       % 其他情况
   f = x^2*y^3;            % 赋值
end
```

当逻辑表达式比较冗长的时候，建议用户引入中间临时逻辑变量来替换复杂的逻辑表达式，比如：

```
if (x+y*sc>Imax)&(y-ks*B<Imin)&mod(N+k*P,2);
…       % 满足条件时执行的程序段
End
```

上面的程序可以等价地写为下面的格式：

```
C1 = (x+y*sc>Imax);
C2 = (y-ks*B<Imin);
C3 = mod(N+k*P,2);
```

```
if C1&C2&C3
...         % 满足条件时执行的程序段
End
```

如此替换之后，用户可以直接根据语句阅读程序，而且便于程序的调试。

在 if ... else ... end 结构中，应该把频繁发生的情况写在 if 部分，例外情况写在 else 部分，这样可以增加程序的可读性。

一般，if...elseif ... else... end 语句可以转化为如下格式：

```
f = exp1.*(cond1)+ exp2.*(cond2)+...
```

其中 exp1 和 exp2 等是函数的计算表达式，cond1 和 cond2 等是条件表达式。

比如例 5-1 中的表达式还可以写为下面的形式：

```
f=[x.^2.*y].*(x<=0&y<=0)+[x.*y.^2].*(x<=0&y>=0)+[x.^2.*y.^2].*(x>=0&y<=0)+x.^2
.*y.^3.*(x>=0&y>=0);
```

因为 x=0 和 y=0 的情况包含在其他 4 种情况中，即利用“ >=”和“<=”代替“>”和“<”。同时上面的计算表达式和条件表达式组合的形式还具有支持向量计算的优点，而使用条件语句格式不支持向量计算，编程的时候需要结合循环结构来逐点计算。

这里配合一些实际数据来测试上面的程序，相关语句如下：

```
t = linspace(-2,2,40);       % 生成[-2,2]范围内的等间隔向量序列
[x,y] = meshgrid(t);         % 矩形范围内的坐标数据
f=[x.^2.*y].*(x<=0&y<=0)+[x.*y.^2].*(x<=0&y>=0)+[x.^2.*y.^2].*(x>=0&y<=0)+x.^2
.*y.^3.*(x>=0&y>=0);
mesh(x,y,f);                 % 绘图
xlim([-2,2]);                % 设置 x 坐标轴范围
ylim([-2,2]);                % 设置 y 坐标轴范围
colormap([0,0,0])            % 设置曲面为黑色网格
```

上面程序执行后得到的图形如图 5.1 所示。

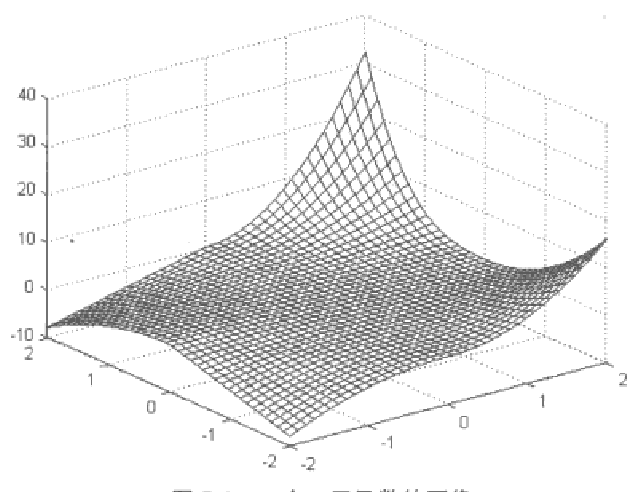


图 5.1　一个二元函数的图像



计算表达式和条件表达式组合的形式需要在用户熟练的情况下调用，其可读性比较差，优点是支持向量计算，可以提高程序的执行速度。

5.2 switch 语句

switch ... case ... end 和条件语句一样，是一种分支结构。switch 语句在编写程序时要比 if 语句简单些，也容易理解，这种格式使用于分支情况比较多的场合，其调用的基本格式为：

```
switch switch_expr,
   case case_expr,
      statement,
   case {case_expr1, case_expr2, case_expr3,...}
         statement,
   ...
   otherwise,
      statement,
end
```



switch_expr 是检测的变量名。case_expr 是 switch_expr 可能的取值，statement 是满足条件时相应的操作程序段。otherwise 用于处理除上述 case 关键词所列情况之外的情形。当 switch_expr 和 case_expr 相等时将执行相应的语句。其中 case_expr，case_expr1，case_expr2 等可以是 double 型或者 char 型数据。switch 和 end 之间的语句按先后顺序检查，直到遇到满足条件的情况时结束 switch 语句而运行后面的程序内容。用户在利用 switch 语句的时候分类条件应该是完备的，即包含所有可能发生的情况。使用 otherwise 可以补充 case 未列及的情况。

例 5-2： 利用 switch ... case ... end 结构实现下述功能：当 n=0 时输出"任务未完成"，当 n=1，2，3 时输出"任务基本完成"，当 n 等于其他数值时输出"任务很好完成"。

分析： 情况共分为 3 种，使用 2 个 case 和 otherwise 即可实现。字符串可以不加分号显示于命令窗，变量 n 利用 input 函数来输入。实现程序如下：

```
clear;clc;                    % 清空变量，清屏
n = input('请输入数字: ');     % 输入数字
switch n
   case 0
      sh = '任务未完成'
   case {1,2,3};
      sh = '任务基本完成'
   otherwise
      sh = '任务很好完成'
end
```

执行上述程序将会在命令窗中输出：

```
请输入数字：
```

输入 2 后会得到相应的结果：

```
sh =任务基本完成
```

5.3 循环结构

循环结构可以用来重复计算某一任务，在 MATLAB 中提供了 for 和 while 两种实现循环函数的结构，它们分别实现固定和不定的次数计算。本节详细介绍这两种循环结构。

for 循环可以执行固定或者预定次数的循环计算，其调用格式为：

```
for k=1:N;
    statement;
end
```

参数说明：k 是循环的指标，可以是自然数也可以是 double 型数据。一般循环结构是用来计算某一向量或者矩阵型数据，而 k 就是相应的索引脚标。

如果计算矩阵型数据，则需要利用两层 for 循环结构，即：

```
for m=1:M;
    for n=1:N
        statement;
    end
end
```

参数说明："1:N" 和 "1:M" 是向量结构，用户也可以直接输入 1 个向量名，N 和 M 是程序计算的范围。计算顺序可以从小到大，也可以从大到小（此时应该用 "N:-1:1" 来代替 "1:N"）。statement 表示每一步中要完成的计算任务。

在一些特殊场合，可能需要利用多层循环，用户根据上面的两层循环类推即可得到多层循环的结构。在调用循环结构的时候，循环每一步都计算的量可以放到循环结构前面进行计算，如此可以节省不必要的计算。

如果用户在循环计算的过程中要求满足条件时中止循环计算，可以在 for 循环里面加入 break 函数来中止程序，使用格式如下：

```
for k=1:N;
    statement;
    if cond1
        break;
    end
end
```

其中 cond1 用来检查是否满足条件。

与 break 函数类似的一个函数是 return 函数。这个函数可以使当前正在运行的函数正常结束，同时返回调用它的函数继续运行后面的程序，或者返回到调用它的环境，如命令窗和 MATLAB 程序文件中。该函数通常用于函数文件中。用户可以对输入的参数进行判断，如果参数不符合要求，就调用 return 语句中止当前程序的运行，并返回到调用它的函数或环境，这样可以省去一些不必要的计算量。

例 5-3：从自然数 1 开始累加，加数为自然数的质数因子最小数，直至累加和达到 100 时停止累加，返回累加和于停止的位置。

分析：利用 for 循环来完成这个计算任务。其中质数分解可以利用函数 factor 来计算，并利用

min 函数来计算最小值。程序的结束控制通过 if 语句调用 break 函数来实现。因为在循环计算之前不清楚需要计算到哪一个自然数停止，因此这里预设程序计算到自然数 100。实现程序如下：

```
S = 0;                % 初始化累加变量
for k = 1:100;
   F = factor(k);      % 对 k 进行质因数分解
   Fm = min(F);       % 获得所有因数中的最小值
   Fn(k) = Fm;        % 记录最小质数
   S = S+Fm;         % 累加求和
   if S >= 100;       % 检查 S 是否大于等于 100
      break;        % 满足条件停止程序
   end
end
k,S,Fn
```

上述程序输出如下：

```
k =    19
S =   100
Fn =  1     2     3     2     5     2     7     2     3     2    11     2    13     2
3     2    17     2    19
```

可见截止的自然数是 19，累加和是 100。Fn 记录了相应的最小质数。

为了让 for 循环计算得快些，在循环之前需要给数组预先分配一个空间。比如例 5-3 中每执行一次循环变量 Fn 的 size 就增加 1，这样 MATLAB 需要花费时间对变量 Fn 分配内存来存储数据。为了避免这个操作浪费时间，可以在 for 循环之前增加如下一条语句：

```
Fn=zeros(1,19);
```

这样就为变量 Fn 预先分配了内存，在每次计算中只是对 Fn 相应位置的值进行更换，而不必考虑增加内存的事情。

如果对向量或者数组内所有元素执行的计算任务都是一样的，可以利用支持矩阵计算的计算关系或者函数来并行计算，避免用 for 循环来重复计算。比如 t=0:.1:1，计算 sin(t*pi)/[2+cos(2*t*pi)] 时，可以考虑利用下面的语句：

```
t = 0:.1:1;
y=sin(t*pi)./[2+cos(2*t*pi)];
```

这样程序将会执行得很快，用户输入的内容也会减少，但是程序的可读性会降低。希望用户在熟练以后才使用这种简化的形式，并进行详细的注释以备日后方便查询和其他用户阅读学习。在后面的“程序加速”一节会更详细地分析这种形式的编程风格。

与 for 循环不同的是，while 循环将以不确定的次数执行循环结构，其调用格式为：

```
while expression
   statements
end
```

参数说明： expression 是一个表达式，用来控制程序是否执行，一般情况下是一个逻辑表达式或多个逻辑表达式。statements 是满足条件时每一步循环操作的内容。当 expression 的结果为真时，在 statements 里面的程序内容将被执行。大多数情况下，表达式 expression 的所得结果为 1

个标量值，对于数组变量同样有效。当 expression 结果为 1 个数组时，要求所有元素都为真时才执行程序，否则程序结束。

例 5-4：获得 100 以内所有加法表达式，其中加数有两个，它们是在 1~99 之间的数，同时限制第 1 个加数小于第二个加数。要求把所有加法算式在命令窗中显示。

分析：这个问题考虑利用 while 循环结构来计算。利用函数 disp 来把字符串组成的表达式显示到命令窗中。程序如下：

```
a = 1;    % 初始化加数 a
b = 2;    % 初始化加数 b
N = 0;    % 加法算式个数的计数器，这里初始化为 0
while a<100;
    if a+b<100.5;                                          % 验证是否溢出
        disp([num2str(a),'+',num2str(b),'=',num2str(a+b)])      % 输出加法算式
        b = b+1;                                           % 变化 b
        N = N+1;                                           % 增加算式个数
    else
        a = a+1;                                           % 变化 a 的值
        b = a+1;                                           % 变化初始化 b 的值
    end
end
N
```

程序计算输出如下：

```
1+2=3
1+3=4
...       % 这里略去以下等式
```

输出 N 的值为：

```
N =        2450
```

等价地利用两重循环结构来计算这个问题，相应程序如下：

```
a = 1;    % 初始化加数 a
N = 0;    % 加法算式个数的计数器，初始为 0
while a<100;
    b = a+1;
    while a+b<100.5;                                       % 验证是否溢出
        disp([num2str(a),'+',num2str(b),'=',num2str(a+b)])      % 输出加法算式
        b = b+1;                                           % 变化 b
        N = N+1;                                           % 增加算式个数
    end
    a = a+1;                                               % 变化 a
end
N
```

其中外层循环是变化 a，里面一层循环是变化 b。两层循环输出的结果和一层循环输出的结果一样，这里不再显示。

和 for 循环一样，存在数组定义时，需要预先定义出数组的内存空间，这样可以避免在程序计算过程中出现动态内存浪费的问题。

前面介绍的程序结果，如在 if，switch，for 和 while 结构中和 MATLAB 程序中，关键词都是上

下对齐，如 if 和 end，for 和 end。在输入的时候，这些关键词之间相互匹配，并显示为蓝色。在大型程序中，出现因结构引起的错误是比较难查找的，但是利用 MATLAB 自带的 Edit 编辑器会自动缩进语句，并以不同的颜色显示不同作用的字符，便于查找错误。因此希望用户输入程序内容时按默认的标准格式进行。

5.4 递归结构

当一个计算任务和它自身的规则存在关系时，就可能涉及递归关系了。在所有程序结构中，递归结构是很难理解和掌握的。比如方形几何中的自相似性就是一个典型的例子，局部和整体之间的相似性拓展可以得到美丽的方形图像。程序设计也会因为使用递归变得简单，但是不容易理解，可读性差一些。

在 MATLAB 中递归结构需要写到函数文件中，函数在其内部调用自身。简单地说，递归结构就是函数自身的调用。下面以阶乘函数的建立来介绍递归结构。代码如下：

```
function y=fac(n);
if n==0;
   y=1;
else
   y=n*fac(n-1);     % 自身调用
end
```

把上述内容存为 fac.m 文件即可实现阶乘的计算。这里，在 fac.m 文件中调用了 fac 来计算 n-1 时的值，拿 n=5 来说，其计算过程依次是 5*fac(4)→4*fac(3)→3*fac(2)→2*fac(1) →1fac(0)，而 fac(0)由 if 后面的语句定义，这样就可以得到 fac(5)的值了。



在 MATLAB 中，已经提供了 factorial 和 gamma 函数可以用来计算阶乘。

下面再举一个二值图形处理的例子。

例 5-5：计算任务是在 1 个字符图像上面删去 1 个连通的文字。初始的图片如图 5.2 左图所示（相应图片在光盘中，文件名是 words.bmp，位于 Ch05 文件夹），通过程序实现删去其中的“工字”。

分析：图中分离的几个字符都是连通在一起的，只要抓住一点，就可以“顺藤摸瓜”得到全部的“黑点”。对于二值图像对应的矩阵，其中只有 2 个数值，即 0 和 1。而每个矩阵元素周围有 8 个近邻，如果当前点为黑点（对应像素值等于 0），那么把该点变为白色，同时处理周围的 8 个像素。因此可以建立一个递归过程来处理这个问题。





处理之前的图像　　删去一个字的图像

图 5.2 程序执行的结果

相应的实现程序如下：

```
function [A,xn,yn,x,y] = pickup(A,xn,yn,x,y);
if A(x,y) == 0;
    xn = [xn,x];                                   % 满足条件后把位置坐标 x 记录到 xn
    yn = [yn,y];                                   % 满足条件后把位置坐标 y 记录到 yn
    A(x,y)=1;                                      % 把黑点变为白点
    [A,xn,yn,x,y] = pickup(A,xn,yn,x+1,y);              % 处理第 1 个近邻点
    x = x-1;                                       % 把 x 的值恢复到原来的位置
    [A,xn,yn,x,y] = pickup(A,xn,yn,x-1,y);              % 处理第 2 个近邻点
    x = x+1;                                       % 把 x 的值恢复到原来的位置
    [A,xn,yn,x,y] = pickup(A,xn,yn,x,y+1);              % 处理第 3 个近邻点
    y = y-1;                                       % 把 y 的值恢复到原来的位置
    [A,xn,yn,x,y] = pickup(A,xn,yn,x,y-1);              % 处理第 4 个近邻点
    y = y+1;                                       % 把 y 的值恢复到原来的位置
    [A,xn,yn,x,y] = pickup(A,xn,yn,x+1,y+1);            % 处理第 5 个近邻点
    y = y-1;                                       % 把 y 的值恢复到原来的位置
    x = x-1;                                       % 把 x 的值恢复到原来的位置
    [A,xn,yn,x,y] = pickup(A,xn,yn,x-1,y-1);            % 处理第 6 个近邻点
    y = y+1;                                       % 把 y 的值恢复到原来的位置
    x = x+1;                                       % 把 x 的值恢复到原来的位置
    [A,xn,yn,x,y] = pickup(A,xn,yn,x-1,y+1);            % 处理第 7 个近邻点
    y = y-1;                                       % 把 y 的值恢复到原来的位置
    x = x+1;                                       % 把 x 的值恢复到原来的位置
    [A,xn,yn,x,y] = pickup(A,xn,yn,x+1,y-1);            % 处理第 8 个近邻点
    y = y+1;                                       % 把 y 的值恢复到原来的位置
    x = x-1;                                       % 把 x 的值恢复到原来的位置
end
```

把上述内容存为 pickup.m，即可实现删除某文字的功能，其中在函数 pickup 内调用了该函数。相应的调用程序如下：

```
A = imread('words.bmp');                 % 载入图片文件为 1 个矩阵
x = 40;                                  % 设定 1 个 x 初始点，作为"顺藤摸瓜"的起点
y = 40;                                  % 设定 1 个 y 初始点，作为"顺藤摸瓜"的起点
xn = [];                                 % 初始化坐标点集 xn
yn = [];                                 % 初始化坐标点集 yn
subplot(121);                            % 开一个子图窗口
imshow(A,[]);                            % 绘图
xlabel('处理之前的图像');                 % 标注
[A,xn,yn,x,y] = pickup(A,xn,yn,x,y);     % 调用 pickup 进行处理
subplot(122);                            % 开另外一个子图窗口
imshow(A,[]);                            % 绘图
xlabel('删去 1 个字的图像');              % 标注
set(gcf,'Color',[1,1,1]*0.92);           % 背景色设置
```

上述程序语句执行后可以得到如图 5.2 右图所示的图形。通过选择不同的初始点，还可以删去其他字符。



初始点的像素值应该等于 0，否则程序不能删除图像上的内容。

当递归次数超过 500 次时，MATLAB 会提示出错，并告诉用户增加最大递归次数设置。有的

时候是递归程序的问题，需要做相应的修改。当计算量比较大时，需要特别地增加最大递归次数，相应用法如下：

```
set(0,'RecursionLimit',N);
```

5.5 人机交互函数

MATLAB 提供了 input，keyboard，echo 和 profile 等函数来实现人机交互过程，下面详细介绍这些函数的使用方法。

函数 input 可以让用户在命令窗中输入相关的参数，其调用格式为：

```
R = input('expression');            % 格式 1
R = input(' expression','s');       % 格式 2
```

参数说明：expression 是对变量 R 进行说明，比如某某参数。参数 s 用于指定输入内容是字符串。执行 input 函数之后会在命令窗中出现 R= expression，在 expression 后面用户可以输入合法的内容，如 R 可能的取值或者 workspace 里已经存在的变量名。如果输入的内容错误，将会提示出错，同时要求用户继续输入。

如执行语句 R = input（'折射率:'），首先要把这个语句输入到命令窗：

```
>> R = input('折射率=')
```

如果输入 1 个不存在的变量 A，如下：

```
折射率=A
```

将会出现如下出错信息：

```
??? Error using ==> input
Undefined function or variable 'A'.
```

并提示继续输入（如果输入变量名是存在的，MATLAB 将通过并继续执行后面的程序）：

```
折射率=
```

可以输入正确的数值，如：

```
折射率=1.617
R =    1.6170
```

在输入过程中，用户可以输入向量或者矩阵，按回车键后 MATLAB 认为输入结束，并把输入内容赋给变量 R。

如果是用户自己编写和使用的 M 文件，应该直接在程序中赋值及相应地写上注释，这样比利用 input 函数方便，因为执行程序时不必逐个输入参数而节省了时间。在给其他用户演示或者使用时可以考虑利用 input 函数，因此用户需要根据不同情况来选用这个函数。

函数 keyboard 把计算机作为一个函数文件来调用，并放入 M 文件中，这个特性对调试或正在运行期间修改变量很有帮助。当执行该函数时，MATLAB 将暂时停止运行程序，并使计算机处于等待键盘输入状态。一旦处理完需要的工作后，输入 return 并按回车键，程序将继续运行。在 M 文

件中使用这个函数，对调试程序非常有用。类似的函数是 pause，该函数用于使程序暂停运行，等待用户按任意键才继续，其中 pause(n)将暂停 n 秒钟后继续运行（n 可以是小数）。

函数 echo 通常用来控制 M 文件在执行过程中是否显示。在正常的执行过程中，M 文件是不会显示在命令窗中的。在一些特殊场合，比如需要对 M 文件进行调试和演示时，则需要 M 文件执行的每条命令都显示出来，用函数 echo 就可以实现。其调用格式为：

```
echo on;    %格式 1
echo off;        %格式 2
echo;       %格式 3
```



格式 1 用于打开 M 文件执行过程的显示方式。格式 2 用于关闭显示格式，此为默认状态。格式 3 用于切换格式 1 和格式 2。在显示方式下，M 文件是逐行解释执行的，而不是编译后执行的，这样会大大地降低程序执行的速度和效率。除非有必要，比如在调试程序的时候，否则用户最好不要打开 MATLAB 文件的显示方式（这样影响运行速度）。

函数 profile 可以用于详细地查看（或者叫剖析）MATLAB 文件中每步耗费的时间，用户可以看到程序费时的部分。其调用格式如下：

```
profile on
profile off
profile report
profile plot
profile reset
profile done
profile clear
```

参数说明： on 用于打开 profile 函数的剖析功能。off 用来关闭剖析功能。report 用于显示剖析结果。plot 给出一个图形来描绘剖析结果。reset 把消失时间数据恢复为 0。done 关闭剖析功能并清除相应数据。clear 用于清除所有记录的剖析结果。在低一些 MATLAB 版本（如 6.5 版本）中，可能参数有所不同，读者调用时请先仔细阅读帮助文档。

例 5-6： 计算 num2str 函数的剖析过程。

分析： 首先要建立一个简单的程序来调用 num2str 函数，再调用 profile 函数获得剖析结果，其中显示剖析结果数据，并显示相应的图形。相关程序如下：

```
Ns = [randperm(8)]';
profile on                % 打开剖析操作
g = num2str(Ns);          % 把 double 型数据转化为字符串类型
profile report              % 弹出剖析结果窗口
profile viewer            % 停止剖析并显示结果于 IE 浏览器中
profsave(profile('info'),'profile_results');   % 保存剖析结果于 profile_results 文件夹下
profile off                % 关闭剖析操作
```

执行上述程序后会弹出如图 5.3 所示的窗口。用户可以从这个窗口中查看相关函数执行的时间，同时在当前路径下得到 profile_results 文件夹，并存储了相应的 IE 文件，如图 5.4 所示。

Profiler

File Edit Debug Desktop Window Help

Start Profiling Run this code:

Profile Summary

Generated 21-Jul-2008 17:31:28 using cpu time.

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
num2str	1	0.013 s	0.013 s	
int2str	1	0 s	0.000 s	
log10	1	0 s	0.000 s	

图 5.3 函数 profile 测试程序执行时间的窗口

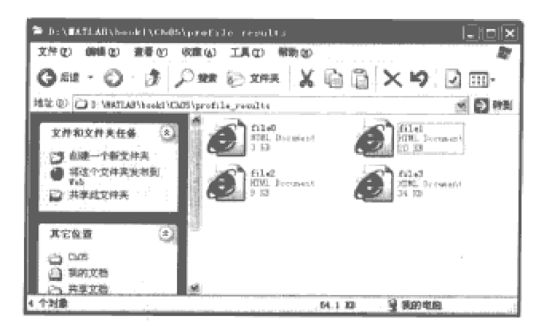


图 5.4 剖析结果存储得到的 IE 文件

5.6 程序加速

本节介绍一些加速程序在执行时的一些方法，这些方法主要是在编程风格上做一些特殊的改进。因为 MATLAB 对于矩阵计算支持得非常好，所以加速程序主要围绕向量化编程。对于提高程序运行速度，这些方法非常有用。希望读者通过本节可以提高程序执行的速度。加速程序需要注意以下几个方面。

◆ 只有使用这样的数据类型，MATLAB 才会对其加速：logical，char，int8，uint8，int16，uint16，int32，uint32，double，而语句中如果使用了非以上数据类型时则不会加速，比如：numeric，cell，structure，single，function handle，java classes，user classes，int64，uint64 和稀疏矩阵。在运行时数据类型分析和处理方面，由于 MATLAB 中的变量可以非常灵活地进行类型转换和数组大小的调整，因此 MATLAB 必须具备强大的类型检查功能，MATLAB 为此项任务需要付出时间代价。在运行过程中，数据类型分析根据一个原则来提高类型分析的效率，即若一行 MATLAB 语句被执行两次，则第二次执行的语句中的变量，其数据类型和大小极有可能和前一次相同，那么代码翻译工作就可以只做一次，即 MATLAB 可以“偷懒”一次。

◆ MATLAB 不会对超过三维的数组进行加速。

◆ 当使用 for 循环结构时，只有遵守以下 3 条规则才会被加速：

- for 循环的范围只用标量值来表示。
- for 循环内部的每一条语句都要满足上面第 1 条和第 2 条规则，即只使用支持加速的数据类型，只使用三维以下的数组。
- 循环结构内只调用了内建函数（build-in function）。

◆ 当使用 if (elseif)，while 和 switch 时，其条件测试语句中只使用标量值，才加速运行。

◆ 不要在一行中写入多条语句，这样会减慢运行速度，而且不利于程序的调试。希望用户不要编写如下语句：

```
x = A.name; for k = 1:20000, y(k) = sin(x(k)); end;
```

因此，规范地编写程序是非常重要的。其中道理不难解释，程序越是规范，编译器需要猜测的事情就越少，效率自然越高。

◆ 当某条语句改变了原来变量的数据类型或形状（大小、维数）时，将会减慢运行速度。

◆ 对于复数常量的书写，应该这样使用复常量：x = 7 + 2i，而不是像这样使用：x = 7 + 2*i，后者会降低运行速度。

◆ 在程序结构上，尽量避免使用循环。

◆ MATLAB 是一种矩阵语言，针对向量和矩阵操作而设计。向量化设计 M 文件可以利用矩阵语言的特点，一些 for 循环和 while 循环可以等价地写为 1 个向量或者矩阵操作。改进这样的状况有两种方法：

 ● 尽量用向量化的运算语句来代替循环结构。比如下面的程序：

```
i=0;
for t = 0:.01:10
i = i+1;
y(i) = sin(t);
end
```

可以替换为：

```
t = 0:.01:10;
y = sin(t);
```

因此速度将会大大加快。最常使用向量化技术的函数有：All，diff，permute，permute，reshape，squeeze，any，find，logical，prod，shiftdim，sub2ind，cums，um，ind2sub，ndgrid，repmat，sort 和 sum 等。



MATLAB 文档中还有这样一句补充：在向量化程序代码的时候还要注意执行加速的情况，用户可以用 MATLABJIT 加速器加速自己的代码。因此，用户需要根据实际情况来确定向量化自己的程序。

 ● 在必须使用多重循环时，如果两个循环执行的次数不同，则程序最好在外环执行循环次数少的，内环执行循环次数多，这样可以显著提高速度。有时还可以考虑用较少循环的重数，比如例 5-4 中给出的两种实现程序，读者可以考虑用设计程序结构来改变循环的重数。

◆ 对于数组操作需要注意以下两点：

 ● 预先分配矩阵内存空间，即事先确定变量的大小和维数。这一类的函数有 zeros，ones，cell，struct 和 repmat 等。在计算时只是更换相应位置的值即可，省去了分配内存的时间。还有一点，在 MATLAB 中变量传递是按照复制方式，所以要尽量避免使用多个子函数，或者在调用函数时避免传递给 size 很大的变量。如果多个函数之间需要同时使用大变量，可以定义成全局变量（global），由此可以极大地提高程序的执行速度。

 ● 当要预分配 1 个非 double 型变量时，使用 repmat 函数可以加速，如将以下代码：

```
A = uint8(zeros(100));
```

换成：

```
A = repmat(uint8(0), 100, 100);
```

当需要扩充1个变量的大小和维数的时候，可以调用函数 repmat 来实现。

◆ 在函数选用上可以增加程序的计算时间，这里涉及 MATLAB 自带的内建函数和用户自己定义的函数文件，具体关系如下：

- 优先使用 MATLAB 内建函数，可以将耗时的循环编写进 MEX-File 中，以达到加速的目的。
- 使用用户函数文件而不是脚本文件。把脚本文件转化为函数文件的办法是在程序最开始处加入下面的语句：

```
function outputN = filename;
```

因为在脚本文件中输入的参数都已经存在，而不必再增加参数，所以用户可以把需要输出的参数写到 outputN 部分，多个参数可以写为[output1, output2,...]。关于函数文件和脚本文件的详细介绍，读者可以参阅第6章内容。

改进程序整体结构，是解决问题的最好方法，思路如下两个方面：

◆ 改用更有效的算法。

◆ 采用 Mex 技术，或者利用 MATLAB 提供的工具将程序转化为 C 语言、Fortran 语言。

但是在版本 R13 以后，MathWorks 声称已经完成所有内建函数及工具箱的加速。同时 JIT（Just-In-Time）代码生成技术从根本上改变了原来 MATLAB 程序代码的执行过程。原来版本的 MATLAB 代码的处理方式是，先翻译成 p-code（P 码文件），再用 p-code 解释器执行。而 JIT 技术可以将许多 MATLAB 程序直接转化为机器指令，省去了中间的解释工作所耗费的时间。该技术在 Intel X86 体系的 Linux 和 Windows 上有额外优化的余地。JIT 加速器详细内容可以参阅 http://www.mathworks.com/company/newsletters/news_notes/may03/profiler.html。

Mex 技术对提高程序运行速度已无用，其作用只是可以增加程序的可移植性而已，有时反而还会出现转换后函数的运行速度变慢的现象。

在加速程序的时候，程序可能变得不容易阅读，此时用户需要进行详细的注释，同时也需要积累一些关于 MATLAB 内建函数执行速度的知识，用执行效率高的函数可以极大地加快程序执行。对于一些有用的工具化程序段，可以编写为函数文件以作为自己的工具函数使用，这样不仅可以加速程序，还方便编程，节省了程序内容输入方面的工作量。

5.7 程序注释

在 MATLAB 中，注释内容放在百分号“%”后面，对程序中相关语句的功能做注释，供用户记录程序，便于日后回忆程序的整体功能。需要注意的是在换行符号“...”后面写入注释会引起错误，因此需要在语句结束后写入注释内容。MATLAB 在编译的时候会忽略百分号后面的内容，把从百分号开始到本行末尾的内容作为注释部分。当需要大段注释语句的时候，用户可以先选择要注释的部分，再在右键快捷菜单中选择 Comment 命名就可以注释相应的语句了。如果想取消注释，可以在右键快捷菜单中选择 Uncomment 命名。利用大段语句的注释和取消注释可以控制是否执行程序，这一特点在调试或者保存不用的可选程序段方面是非常方便的，有的内容可以作为以后的参考。

有的时候在语句后面增加注释内容还是不能很好地记录程序的功能，用户可以使用其他文档，如 Word 文档来书写详细的记录，同时可以结合公式和图形来记录。

5.8 常见错误的调试

本节介绍一些程序调试方面的知识，可以方便用户快速建立需要的程序。如果是用户自己编写一个全新程序，建议应适当地对程序进行编译，检查是否出现错误，发现错误应及时修改。如果是修改别人的程序，需要自上而下地调试程序，并根据提示对程序做相应修改，此时程序的流程、参数取值及要实现的任务是非常重要的，如果没有这些知识，程序是很难调试正确的。一般地，程序的错误可以分为语法错误和非语法错误。下面详细介绍这两种错误的调试方法。

5.8.1 语法错误

语法错误是指程序的错误可以由 MATLAB 通过编译显示出来。这样的错误用户可以根据提示进行修改，一般情况下这类错误是很容易修改的。下面将列举一些常见的语法错误。

5.8.1.1 索引超出规定

因为索引超出规定范围而引起的错误，这是一种在编写程序时经常遇到的错误。下面是一段存在这样错误的程序段：

```
M=5;          % 参数赋值
N=6;          % 参数赋值
Ma=rand(M,N); % Ma 是随机矩阵
for n=1:N;
    for m=1:M;
        Mb(n,m)=[Ma(n,m)+1]*sin([n/M+m/N]*pi); % 计算矩阵 Mb 的值
    end
end
```

执行上述程序将会出现如下提示：

```
??? Attempted to access Ma(6,1); index out of bounds because size(Ma)=[5,6].
Error in ==> debug_list at 9
        Mb(n,m)=[Ma(n,m)+1]*sin([n/M+m/N]*pi);
```

在“???”之后的一行文字是告诉用户出现错误的信息，其具体指出了调用 Ma(6,1)时超出矩阵 Ma 的索引范围，在 6.5 版本中则给出较为含糊的提示（只是给出超出范围的提示）。后面两行的提示信息表明出现错误的位置（在程序中的行数）和相应语句。在位置提示的下面有一条下画线，表示是 1 个超链接，用户单击后可以直接把光标移到当前 M 文件出错的位置。改正这个错误的办法是把语句“Mb(n,M)=[Ma(n,m)+1]*sin([n/M+ m/N]*pi);”变换为“Mb(n,M)=[Ma(n,m)+1]*sin([n/M+m/N]*pi);”就可以了。

5.8.1.2 矩阵乘法中两个矩阵的尺寸不合法

矩阵乘法中两个矩阵的 size 不合法，这种错误一般是因为用户疏忽了矩阵的 size 关联而引起的，比如下面的例子：

```
A = magic(5);
B = rand(6,5);
C = A*B;
```

执行上面例子之后将会显示如下出错信息：

```
??? Error using ==> mtimes
Inner matrix dimensions must agree.
Error in ==> debug_list at 23
C = A*B;
```

这里 MATLAB 提示用户矩阵的维数需要一致，同时指出了出错位置和相关语句。调整矩阵的 size 就可以改正这样的错误了。

5.8.1.3 警告提示信息

警告提示信息的处理，在一些计算或者某些函数的调用中可能会出现警告提示，比如：

```
Z=1e52*rand(200);   % 生成 1 个动态范围大的矩阵
axes; % 打开 1 个空的坐标轴
mesh(Z) % 绘图
```

执行上述命令会出现 1 个空的坐标轴并出现如下提示：

```
Warning: Axis limits outside float precision, use ZBuffer or Painters instead. Not
rendering
```

出现这样的警告是因为数据范围超出了 MATLAB 允许的范围，改正的办法是：在使用 mesh 命令之前，使用下面的语句就可以了。

```
set(gcf,'Renderer','ZBuffer');
```

或者：

```
set(gcf,'Renderer','painters');
```

如果在程序中出现这样极大数据或者极小数据，可能是计算公式编写存在问题，而导致出现这么大的数。在处理警告之前，用户应该检查结果是否满足物理现实。

对于 MATLAB 弹出的警告信息，有的可能不影响程序的执行，但是用户应该仔细核查自己的程序，避免警告出现。

5.8.1.4 输出矩阵大小的限制

某些函数对于输出矩阵的 size 做了限制，如 zeros，rand 和 ones 等函数。比如下面的语句：

```
M=rand(100000,10000);
```

MATLAB 会提示用户如下信息：

```
??? Maximum variable size allowed by the program is exceeded.
```

表示 size 参数超出了 MATLAB 允许的范围，把数值变小即可。在程序设计中可以考虑用多个小矩阵来组合成为大的矩阵。

5.8.1.5 表达式非法

表达式非法也会引起错误，这类错误是因表达式中的符号错误而引起的，如：

```
x=1;
y=exp(-0.5x);
```

将会出现如下提示：

```
Error: Unexpected MATLAB expression.
```

这里在 0.5 和 x 之间缺少 1 个运算符号，加上“+”、“-”“*”和“/”等就可以正常运行了。

```
x=0:0.01:5;
y=x^2;
```

执行上面的语句会出现下面的提示：

```
??? Error using ==> mpower
Matrix must be square.
```

只有用方矩阵的时候乘方运算“^”才是合法的，而这里 x 是 1 个向量，只能写为点运算，即“.^”，这样将进行 x 的各元素的对应幂运算。类似的矩阵元素对应相除还可以使用表达式“A./B”。

5.8.1.6 因变量类型错误

因变量类型引起的错误是因为不同数据类型的混用而引起的，如：

```
syms x y            % 定义符号变量
f = inline('x+y');      % 定义内联函数
Iv = int(int(f,x,1,2),y,1,2) % 进行积分计算
```

执行后会提示：

```
??? Undefined function or method 'int' for input arguments of type 'inline'.
Error in ==> debug_list at 26
Iv = int(int(f,x,1,2),y,1,2)
```

就是说，f 的类型不能取 inline，对于 int 函数，被积函数可以选择符号型变量和字符串型变量。

可能出现的错误有很多，这里仅列出经常遇到的几种，用户在实际应用中还需要从不同角度思考如何修改错误，在查找错误原因时需要避免思维定势。对于程序执行过程中出现的错误和提示，用户可以先根据提示进行修改。此外还可以考虑在网络上查找答案，或者找熟练的人帮忙。在问题得到解决后建议用户做一下记录，便于日后解决类似问题。

5.8.2 非语法错误

非语法错误是指调试的时候，程序正常运行而不会出现错误提示，但是程序结果不正确的情况。这类错误很难改正，一般是通过用户自己的专业知识背景来检验输出结果是否合理。这就要求用户具有较多的经验，才有可能发现并改正程序中的问题，其修改过程可能需要花费很多时间。建议用户按下列顺序来修正程序：先检查参数赋值是否正确，再检查程序结构的正确性（如循环和分支结构），最后分段程序，逐段排查。如果一时无法发现错误，建议用户重新建立程序流程来编写新的

程序。下面具体介绍两种典型情况。

5.8.2.1 语句顺序不当

因为语句顺序不当而引起的错误。比如例 5-4 中的程序写为：

```
a = 1;    % 初始化加数 a
b = 2;    % 初始化加数 b
N = 0;    % 加法算式个数计数器
while a<100;
    if a+b<100.5;                                          % 验证是否溢出
        disp([num2str(a),'+',num2str(b),'=',num2str(a+b)]) % 输出加法算式
        b = b+1;                                           % 变化 b
    else
        a = a+1;                                           % 变化 a 的值
        b = a+1;                                           % 变化初始化 b 的值
    end
    N = N+1;                                               % 增加算式个数
end
N
```

此时计算得到的 N 值为 2549，与正确的值 2450 不一样，但是程序执行过程中不会出现错误提示。此时检验程序是否正确，只能通过另外一个程序版本来进行对比，或者与别人的结果比较，或者变化特殊的参数进行检验。特别是在科学研究中，不但要求得到不出现错误的程序，还要求计算结果准确。专业经验、特殊值时的结果（比如参数等于 0 或者等于 1 等）、公开发表的结果（大多数情况下是正确的）等对于检验程序的正确性都是非常重要的。

5.8.2.2 while 循环的错误

while 循环不停止或者不执行是 while 循环经常会发生的情况，此时程序不出现错误提示，只能通过过程量的变化来查看结果。其中，程序执行不停止可能是结束条件有误或者所考虑的算法不收敛，而程序不执行是结束条件有误或者参数赋值不正确。

```
N = 1               %初始化次数计数器
rand('state',0);      % 初始化 rand 函数状态数
A1 = rand(1,100);   % 第一振幅的初始化
A2 = rand(1,100);   % 第二振幅的初始化
A2 = A2/sqrt(sum(A1.^2)); % 平衡两个信号的能量
P1 = rand(1,100);   % 第一相位的初始化
A1t = 100;          % 过程位相的初始化
while sum(abs(A1-A1t))>1e-4;
    A2p = fft(A1.*exp(P1*i));    % 傅里叶变换
    P2t = angle(A2p);          % 提取第二位相
    A1t = ifft(A2.*exp(i*P2t));   % 逆傅里叶变换
    P1 = angle(A1t);           % 提取第一位相
    N=N+1
End
```

上述程序是 1 个简单的 G-S 位相恢复算法。执行这个程序的时候，程序将不停地执行，表明算法是不收敛的，满足不了预设的精度。为了让程序快速结束任务的计算，还需要做算法改进。

```
Nt=0;          % Nt 用于记录循环的次数
R = 2;         % 欲开方的数
```

```
x1 = 2;            % 初始值
x2 = 3;            % 初始值
while abs(x1-x2)>1e-6;
    x2 = 0.5*[x1+R/x1];         % 迭代公式进行开方计算
    x1 = x2;                    % 赋值
    Nt = Nt+1;                  % 次数累加
End
```

这个程序实现了利用牛顿法计算 $\sqrt{R}$ 的值，当用户执行上面的程序的时候，将会得到 Nt=1，x1=x2=1.5，而与期望的 $\sqrt{2}$ 相差很远，Nt=1 说明程序执行一次就停止了。仔细分析这个程序可以发现，语句 x1=x2 执行之后，使得程序结束条件 abs(x1-x2)>1e-6 成立，因此把 x1=x2 转移到语句“x2 = 0.5*[x1+R/x1];”前面就可以了。

可见对于非语法错误，用户需要反复仔细调试程序，才可能得到正确的程序。有时隐藏的错误是很难发现的，此时还可以考虑发动多人分别考虑这个问题，采用发散思维来挑错。

5.9 小结

本章主要介绍了程序结构方面的知识。程序结构是程序的框架，如同人体中的骨骼，一个好的程序结构可以帮助用户编写和调试程序。本章首先介绍了条件语句，给出了如何建立条件语句和等效改进方法。和条件语句相似的一种格式是 switch 语句，该语句可以带有多个分支情况。循环结构是用户经常接触的，它可以通过 for 语句和 while 语句来实现确定次数和不定次数的循环结构。递归结构是一种自身调用的格式，本章通过两个例子介绍了递归结构的设计与调试。人机交互函数是可以实现用户和计算机互动操作的函数。另外，本章还概括了加速 MATLAB 程序的几个编程技巧，同时描述了程序注释时需要注意的事项及如何更好地利用这一功能，最后讲述了调试程序时需要注意的一些事项。

◆　　◆　　◆

第 6 章 文件处理

本章包括

- **脚本文件** 介绍脚本文件相关知识。
- **函数文件** 介绍函数文件的定义、inline 函数、分段函数、自函数和私有函数。
- **函数文件与脚本文件的优缺点比较**
- **数据文件** 介绍数据文件的读写操作。
- **图片文件** 介绍图片文件的读写操作。
- **视频和音频文件** 介绍这两种文件的读写操作。
- **文件批处理结构** 给出文件批量处理的程序结构。

MATLAB 输入命令的方式大体有两种。一种是在工作空间中直接输入简单的命令，这种方式适应于语句比较简单、输入方便，而且处理问题比较特殊、没有一定的重复性和普遍性、错误处理比较简单的场合。这种方式主要体现了 MATLAB 作为一种“数学演算和图视化工具”的特点。读者在以上的学习中已经体会到这种工作方式的优越性和方便之处，这正是其他数学计算工具和编程语言无法比拟的。然而单有这方面的功能还不足以体现 MATLAB 的强大、完整性和容易使用的特点。在进行大量重复性计算和输入时，单靠直接输入是一件非常烦琐的事情。

而 MATLAB 提供了另一种工作方式，就是 M 文件的编程方式。M 文件的语法类似于一般的高级语言，是一种程序化的编程语言，但 M 文件又具有其自身的特点。它只是一种简单的 ASCII 码文本文件，语法比一般的高级语言都要简单，程序比较容易调试，人机交互性强。MATLAB 是一种解释性的编程语言，即逐句地解释运行程序。MATLAB 在初次运行 M 文件时，要将 M 文件编程代码装入内存中，此过程会极大降低程序的执行速度，但再次运行该程序时便会直接从内存中取出代码运行，又会加快程序的运行速度。MATLAB 的 M 程序是注重数学计算的一种编程语言，直接采用复数矩阵作为运算的单位，因此不论从形式上还是从语法上来说，都还算比较容易维护。此外，由于 MATLAB 是使用 C 语言编写而成的，因此它与 C 语言有着千丝万缕的联系，如果熟悉 C 语言，掌握 M 文件的编写会比较简单。

M 文件可以在任何文本编辑器（如记事本、写字板、Word）中进行编辑、存储、修改和读取。利用 M 文件，可以自编函数，对已经存在的函数进行扩充和修改，因此对 MATLAB 进行二次开发是非常方便的。M 文件按形式可以分为两种，即脚本文件（也称命令文件）和函数文件。两种文件的扩展名都是“.m”。

本章将介绍 MATLAB 自身文件的处理，以及相关数据文件的处理方法，此外介绍文件的批处理方法以便于在处理大量文件时书写程序。

6.1 脚本文件

如果要输入较多的语句，而且要经常对这些命令进行重复输入修改，可以利用脚本文件来简单

且方便地处理。可以将要重复输入的所有语句按先后顺序放到一个扩展名为“.m”的 M 文件里面，每次开启 MATLAB 之后只要输入 M 文件的名称即可执行该文件，此外在 M 文件窗口单击【run】按钮或者按【F5】键也可以执行 M 文件。需要注意的是，M 文件要存放在 MATLAB 的搜索路径下，并且文件名不能和 MATLAB 自带函数（包括内建函数和工具箱里面的函数）重名，否则会产生混淆并导致一些不必要的麻烦。如果当前路径下没有要执行的文件（script_file.m 文件），那么执行该文件（单击【run】按钮或者按【F5】键）时，将会弹出如图 6.1 所示的窗口。

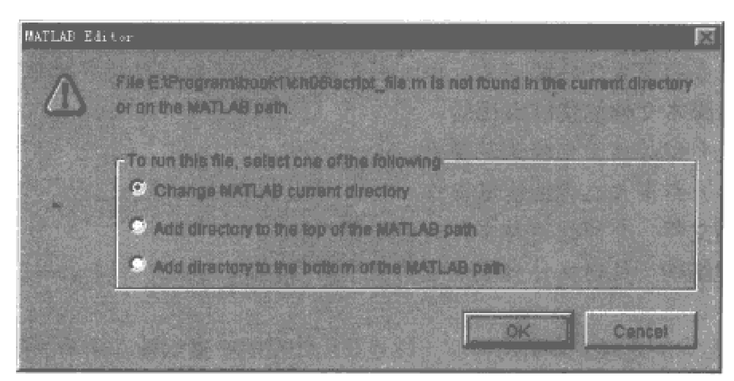


图 6.1　文件不在当前路径时弹出的对话框

单击按钮程序即可执行，同时当前路径切换为文件 script_file.m 所在的路径。如果是在命令窗中输入文件名 script_file 来执行该文件：

```
>> script_file
```

将会出现下面的提示：

```
??? Undefined function or variable 'script_file'.
```

实际上，脚本文件和 DOS 下的批处理命令，与宏命令等有相似之处，MATLAB 对脚本文件的执行等价于从命令窗中按顺序执行文件中所有指令。

函数文件中的语句可以访问 MATLAB 工作空间（workspace）中的所有变量。在脚本文件运行过程中产生的所有变量都等价于直接从 MATLAB 工作空间中建立，因而任何其他的函数文件和函数都可以访问这些变量。这些变量产生后就一直保存在内存中，可以使用函数 clear 来删除工作空间中的特定变量或者所有变量。

在 M 文件编写的时候，文件中的第 1 行注释将会加入到当前路径窗口下的“Description”中，描述该文件的功能，借此可以方便地查找文件。比如在 script_file.m 文件中输入如图 6.2 所示的注释。

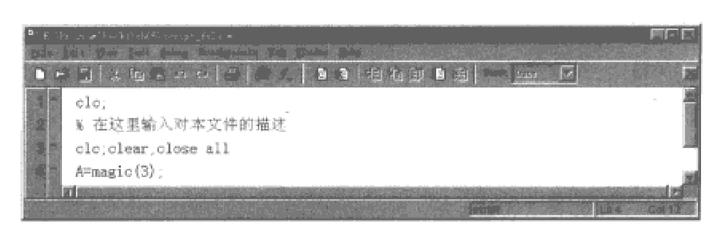


图 6.2　当前路径窗口

在当前路径窗口中将会出现如图 6.3 所示截图中 Description 选中的“在这里输入对本文件的

描述”文字。这样使用者可以用这个方法存储和管理自己的文件。

图 6.3 当前路径窗口截图

如果对第 1 行注释进行修改，MATLAB 将在下次启动后更新注释的内容。如果使用者在 Current Directory 窗口中单击鼠标右键，再在弹出的快捷菜单中选择 Refresh 命令，就可以更新修改后的注释信息，如图 6.4 所示。

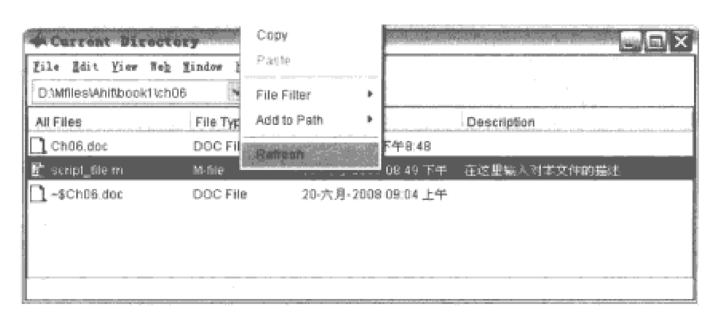


图 6.4 交互式更新注释信息

6.2 函数文件

如果一个 M 文件的第 1 行非注释性语句含有关键词 function，那么这个文件就叫做函数文件。大多数函数文件的第 1 行都是以 function 开头的。函数文件在广义上可以认为是 MATLAB 子函数，其作用与其他高级语言的子函数基本相同，都是为了实现某一功能而定义的。MATLAB 本身提供了数十个工具箱，如信号处理、图像处理、样条分析、神经网络以及金融财政等。这些工具箱扩展了 MATLAB 的功能和应用领域，同时 MathWorks 公司也不断推出新的工具箱，这样很多学科都可以利用 MATLAB 语言来进行科学研究。很多有用的函数就是以函数文件编写的，然而部分函数文件的核心代码是看不到的。只要能确切地知道函数文件的输入和输出关系，就可以放心地使用了。函数文件是扩展 MATLAB 自身功能并对其进行二次开发的强有力的工具。

6.2.1 函数的定义

函数定义的一般步骤是，首先在函数文件的第一行输入：

```
function [y1, y2, …] = function_name(x1, x2, …);
```

其中 function 是关键词，显示为蓝色。y1，y2 等是输出参数，x1，x2 等是输入参数，它们可以是 MATLAB 中不同类型的变量。function_name 是函数文件名，它可以是和函数文件功能相关的英文单词，其首字母只能是英文字母，后面可以是下画线、数字、英文字母等，但是如空格、“-”、

"%"、"+" 等字符是不允许的。

下面是函数文件的例子，该程序计算实现用两点在两个坐标系中的坐标值，建立起坐标系之间的变换矩阵，旋转矩阵 R、平移矩阵 S 及缩放系数，具体内容如下：

```
function [R,S,K]=rotate_shift(x1,y1,x2,y2,X1,Y1,X2,Y2);
% A 是旋转矩阵; S 是平移矩阵; K 是缩放系数
% (x1,y1)和(x2,y2)是第一坐标系的坐标
% (X1,Y1)和(X2,Y2)是第二坐标系的坐标
A = angle([X1-X2]+i*[Y1-Y2])-angle([x1-x2]+i*[y1-y2]);
R = [cos(A),-sin(A);sin(A),cos(A)];
K = abs([X1-X2]+i*[Y1-Y2])/abs([x1-x2]+i*[y1-y2]);
S = [X1;Y1]-R*[x1;y1]*K;
```

在第 1 行下面一般对这个程序的功能及参数的意义进行了说明，然后进行相关计算。MATLAB 中大部分可以看到源程序的函数都是这样的结构。

6.2.2 输入输出参数的控制

有的时候，函数文件的输入和输出变量的个数是不确定的，根据不同的应用，输入和输出参数可能有所变化。MATLAB 中提供了函数 nargin 和 nargout 来计算输入和输出参数的个数。下面介绍这两个函数的使用方法。

函数 nargin 可以返回输入参数的个数，表 6.1 是此函数使用的一个函数文件的例子。

表 6.1 一个函数文件例子

行号	程序内容
1	function y = spview(fun,x0,tol);
2	% 将 fun 按 Taylor 级数展开，要求在 X0 处的误差小于 tol
3	% Example:
4	% fun = 'exp(-2*x^2)';
5	% x0 = 1;
6	% tol = 1e-3;
7	% y = spview(fun,x0,tol);
8	if nargin == 2;
9	tol = 1e-3; % 设置 tol 的默认值
10	elseif nargin ==1;
11	tol = 1e-3; % 设置 tol 的默认值
12	x0 = 1; % 设置 x0 的默认值
13	end
14	fs = sym(fun); % 把 fun 转化为符号表达式
15	fun = inline(fun); % 把 fun 转化为内联函数
16	n = 1; % 初始化展开级数的最高幂次
17	D = inf; % 初始化实时误差
18	while abs(D)>=tol;
19	y = taylor(fs,n); % 计算最高次幂为 n 的 Taylor 展开
20	D = feval(fun,x0)-subs(y,x0); % 更新实时误差 D
21	n = n+1; % 增加展开级数的最高幂次
22	end

在这个程序中，第 2 行描述了这个程序的功能。第 4~7 行是调用函数 spview 的一个例子。第

8~13 行利用 nargin 函数检查输入参数的个数，当输入参数少于 3 个的时候，程序将进行判断，把未定义的变量赋值，如果不定义缺少的变量，下面将会出现变量未定义的错误提示。第 14~18 行定义过程参数。第 18~22 行循环计算 Taylor 分解。计算演示例子的程序有：

```
fun = 'exp(-2*x^2)'; x0 = 1; tol = 1e-3;
y1 = spview(fun,x0,tol)     % 格式 1
y2 = spview(fun,x0)         % 格式 2
y3 = spview(fun)            % 格式 3
```

用 3 种不同输入参数调用 spview 函数，所得结果如下：

```
y1 = 1-2*x^2+2*x^4-4/3*x^6+2/3*x^8-4/15*x^10+4/45*x^12-8/315*x^14+2/315*x^16-4/2835*x^18
y2 =1-2*x^2+2*x^4-4/3*x^6+2/3*x^8-4/15*x^10+4/45*x^12-8/315*x^14+2/315*x^16-4/2835*x^18
y3 =1-2*x^2+2*x^4-4/3*x^6+2/3*x^8-4/15*x^10+4/45*x^12-8/315*x^14+2/315*x^16-4/2835*x^18
```

可见 3 种调用方式，默认参数起了作用，因此得到了相同的结果。在应用中可以根据需要使用不同输入参数个数来调用函数文件。

MATLAB 提供了函数 nargout 来返回输出参数的个数，利用这个函数可以输出不同的参数个数，表 6.2 是一个例子。

表 6.2　可以输出不同参数个数的例子

行号	程序内容
1	function [xp,yp] = Mira_Model(a,b,x,y);
2	% Mira 曲线
3	% Mira_Model(a,b,x,y)
4	N = 12000; % 设置迭代次数
5	xn = zeros(1,N); % 初始化 X 轴坐标
6	yn = xn;　　% 初始化 Y 轴坐标
7	xn(1) = x;　　% 设置 x 的初值
8	yn(1) = y;　　% 设置 y 的初值
9	Fx = a*x(1)+2*(1-a)*x(1) ^ 2/[1+x(1) ^ 2]; % 计算函数 F(x)
10	for k = 2:N;
11	xn(k) = yn(k-1)*b+F(a,xn(k-1)); % 迭代计算 x 的值
12	Fx = a*x(k)+2*(1-a)*x(k) ^ 2/[1+x(k) ^ 2]; % 计算函数 F(x)
13	yn(k) = -xn(k-1)+F(a,xn(k));　　% 迭代计算 y 的值
14	end
15	if nargout == 0;　　% 无输出
16	plot(xn,yn,'k.','markersize',2); % 绘制点图
17	elseif nargout == 1;　　% 一个输出
18	xp = xn;　　% 输出 X 轴坐标
19	elseif nargout == 2;　　% 两个输出
20	xp = xn;　　% 输出 X 轴坐标
21	yp = yn;　　% 输出 Y 轴坐标
22	end



函数文件 Mira_Model 实现了 Mira 模型的迭代计算，该模型的数学描述如 $\begin{cases} x_{k+1} = by_k + F(x_k) \\ y_{k+1} = -x_k + F(x_{k+1}) \end{cases}$，其中 a 和 b 是参数，函数 $F(x)$ 定义为

$$F(x)=ax+\frac{2(1-a)x^2}{1+x^2}$$ 。

函数 Mira_Model 可以有如下 3 种带有不同输出参数的调用方式。

```
Mira_Model(a,b,x,y);                    %无输出直接绘图
xp = Mira_Model(a,b,x,y);               %仅输出 X 坐标
[xp, yp] = Mira_Model(a,b,x,y);         %同时输出 X 和 Y 坐标
```

在程序 Mira_Model.m 中，第 1 行定义函数名、输入参数及输出参数。第 2~3 行是程序功能及调用的注释。第 4~9 行定义参数及相关变量的初始化。第 10~14 行是迭代计算各步 x_k 和 y_k 的数据。第 15~22 行设置输出参数的取值情况。该程序的输出情况是：没有参数时，程序将直接根据迭代数据进行绘图；有 1 个输出参数时，将输出 X 轴坐标的数据；有 2 个输出参数时，输出 X 轴和 Y 轴数据。可以根据实际需要选用不同的输出参数个数。

用下面的程序调用函数 Mira_Model 进行计算，a=0.7，b=0.9998，初始点坐标为(0,12.1)时的 Mira 模型结果如：

```
subplot(121);                           % 第一子图
Mira_Model(0.7,0.9998,0,12.1);          % 调用 Mira_Model，无输出参数
set(gca,'Position',[0.13 0.41 0.327023 0.515]); % 重置坐标轴位置
set(gca,'Fontsize',12);                 % 重置坐标轴字体大小
subplot(122);                           % 第二子图
[x,y]=Mira_Model(0.7,0.9998,0,12.1); % 计算数据
hold on;
plot(x(1:2000),y(1:2000),'k.','markersize',2);              % 1~2000 点用黑色画
plot(x(2001:4000),y(2001:4000),'r.','markersize',2);        % 2001~4000 点用红色画
plot(x(4001:6000),y(4001:6000),'g.','markersize',2);        % 4001~6000 点用绿色画
plot(x(6001:8000),y(6001:8000),'b.','markersize',2);        % 6001~8000 点用蓝色画
plot(x(8001:10000),y(8001:10000),'y.','markersize',2);      % 8001~10000 点用黄色画
plot(x(10001:12000),y(10001:12000),'c.','markersize',2);    % 10001~12000 点用青色画
set(gca,'Position',[0.577977 0.41 0.327023 0.515]);         % 重置坐标轴位置
set(gca,'Fontsize',12);                                     % 重置坐标轴字体大小
```

执行上面的程序得到如图 6.5 所示的图形。左图是直接的绘图结果，从中可以看出整个迭代结果的轮廓；从右图可以观察到不同迭代步骤的位置分布，其中设置了不同颜色绘图，便于观察。

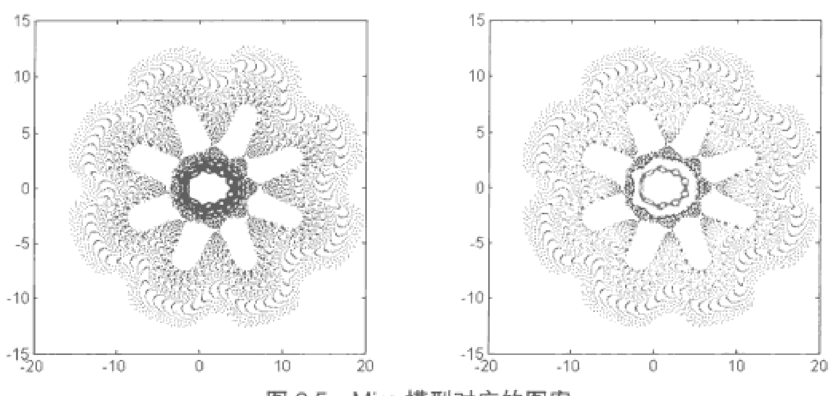


图 6.5　Mira 模型对应的图案

6.2.3　使用内联函数

前面介绍了函数文件的编写方法，这里介绍一种利用 MATLAB 提供的函数 inline 定义的内联函数，这样定义的函数和一般的函数文件用法类似，适用于表达式较为简单的函数。该函数的调用格式为：

```
fun = inline(expr);                   % 格式 1
fun = inline(expr, arg1, arg2, ...);    % 格式 2
```

参数说明： fun 是所定义的内联函数的名称。expr 是表达式对应的字符串，使用者需要正确地输入，否则在利用 fun 进行计算的时候会出现错误提示。arg1，arg2 等是函数中变量的名称。在格式 1 中内联函数 fun 中的变量顺序是按照字母先后顺序排列的。在格式 2 中可以指定变量的顺序。

例 6–1： 利用 inline 定义的函数。

$$f_1(x) = x\sin x - x^2\cos x^2$$

$f_2(x) = x^2/[2+\sin g(x)]$，其中 $g(x) = x + 3\cos(x^3)$

$$f_3(x,y) = x\cos y - y^2$$

对于函数 $f_1(x)$的表达式可以使用函数 inline 如下定义：

```
fun1 = inline('x*sin(x)-x^2*cos(x^2)')
```

执行上面的语句出现如下结果：

```
fun1 =
     Inline function:
     fun1(x) = x*sin(x)-x^2*cos(x^2)
```

但是上面的结果只是支持 x 为单个数字，如运行下面的语句：

```
x = linspace(0,4,100); % 生成数据采样点
plot(x,fun1(x))        % 绘图
```

将会得到如下提示：

```
??? Error using ==> inlineeval
Error in inline expression ==> x*sin(x)-x^2*cos(x^2)
??? Error using ==> *
Inner matrix dimensions must agree.
```

因此需要把 fun1 的定义写为下面的形式：

```
fun1 = inline('x.*sin(x)-x.^2.*cos(x.^2)')
```

此时 fun1 可以支持向量运算。

第 2 个函数 $f_2(x)$可以如下定义：

```
gx = inline('x+3*cos(x.^3)');
fun2 = inline('x.^2./[2+sin(gx(x))]','x','gx');
```

这是一个不同 inline 函数之间调用的问题，下面的 fun2p 是表达式的综合：

```
fun2p = inline('x.^2./[2+sin(x+3*cos(x.^3))]');
```

通过下面的语句，得到的图形如图 6.6 所示。

```
x = linspace(0,4,100);                       % 生成数据采样点
plot(x,fun2(x,gx),'r',x,fun2p(x),'b')         % 绘制两条曲线进行比较
```

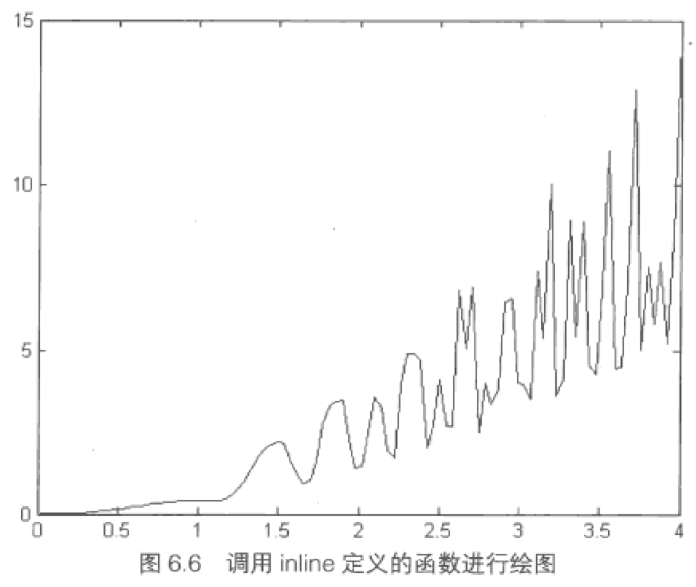


图 6.6　调用 inline 定义的函数进行绘图

可见 fun2 的红色曲线完全被 fun2p 的蓝色曲线所覆盖，即 fun2 和 fun2p 是等价的。可以根据这个方式把较长表达式中重复的部分写为 inline 函数的形式，此外还可以把长的表达式分为两个短的表达式。

第 3 个函数 $f_3(x)$可以按照下面的方式进行定义：

```
fun3 = inline('x*cos(y)-y^2')
```

输出如下：

```
fun3 =     Inline function:
      fun3(x,y) = x*cos(y)-y^2
```

可以看出 fun3 中变量的顺序是按字母顺序排列的，如果用户希望的顺序不是按照字母顺序排列，那么需要指定一下，即：

```
fun3 = inline('x*cos(y)-y^2','y','x')
```

输出如下：

```
fun3 =     Inline function:
       fun3(y,x) = x*cos(y)-y^2
```

在表达式中还可以输入一个向量表达式，比如：

$$F_4(x)=\begin{bmatrix}2+\cos x\\1+\sin x\end{bmatrix}$$

可以如下定义这个向量函数：

```
fun4 = inline('[2+cos(x);1+sin(x)]')
```

按照如下的方式调用 fun4：

```
y = fun4(pi)
```

输出如下：

```
y =    1.0000
       1.0000
```

使用 inline 定义的函数可以在脚本文件中调用，就不必单独使用一个函数文件来定义专门的函数了。因此，所有程序内容都可以编写到一个 M 文件中，便于用户日后的文件管理、保存和传输。当然，如果函数的表达式非常复杂，还是建议用户使用函数文件定义脚本文件。inline 函数如果出现在核心循环，也会使速度下降很多，此时需要考虑用函数文件进行替代，或者直接把表达式写入循环中。在 inline 函数中只能出现函数和参数，要传递一个可变系数 $f_6(x)=\sin(ax)+3e^{-x^2}$，这里 a 是一个参数，可通过下面的方式。

```
a = 1.2;                                              % 对要传递的参数 a 赋值
fun6 = inline(['sin(x*',num2str(a),')+3*exp(-x.^2)'])          % 把参数 a 转化为字符串
写入表达式
y1 = fun6(0.6)                                        % 调用 fun6 函数来计算 0.6 处的值
```

输出：

```
fun6 =     Inline function:
         fun6(x) = sin(x*1.2)+3*exp(-x.^2)
y1 =       2.7524
```

其中，对参数 a 进行 double 型赋值，再利用函数 num2str 转化为字符串。

6.2.4 分段函数

分段函数是一类经常遇到的函数形式，在不同定义区间的表达式不同，而在各区间端点处连续。本小节主要讲述分段函数的定义。比如下面的分段函数：

$$f(x)=\begin{cases} x^2, & 0\le x<1 \\ \cos\left[\pi(x-1)\right], & 1\le x<2 \\ -x^2/(x+2), & 2\le x\le 4 \end{cases} \tag{6-1}$$

可使用表 6.3 所示的函数文件 piecewise1.m 定义这个分段函数。

表 6.3 函数文件 piecewise1.m

序号	程序内容
1	function y = piecewise1(x);
2	if sum(x>=0&x<=4)<0.5; % 检查定义域是否在规定的范围内
3	error('x must be in the interval [0,4]');
4	end
5	if x>=0&x<1; % 检查 x 所属子区间

（续表）

序号	程序内容
6	y = x^2; % 赋值
7	elseif x>=1&x<2; % 检查 x 所属子区间
8	y = cos([x-1]*pi); % 赋值
9	else % 检查 x 所属子区间
10	y = -x^2/[x+2]; % 赋值
11	end

但是上面的程序只能计算当 x 是单个数字的情况，如：

```
y1 = piecewise1(0.5)
y2 = piecewise1(1.2)
y3 = piecewise1(3)
```

而当 x 是一个向量的时候，调用上述函数文件将会出错。表 6.4 给出一种支持向量运算的函数文件编写格式。

表 6.4　一种支持向量运算的函数文件编写格式

行号	程序内容
1	function y = piecewise2(x);
2	if sum(x>=0&x<=4)<0.5; % 检查定义域是否在规定的范围内
3	error('x must be in the interval [0,4]');
4	end
5	y = x.^2; % 计算函数值
6	y(x>=1&x<2) = cos([x(x>=1&x<2)-1]*pi); % 计算函数值
7	y(x>=2) = -x(x>=2).^2./[x(x>=2)+2]; % 计算函数值

在程序 piecewise2.m 中，函数值计算主要是在第 5~7 行。首先利用第 1 区间[0,1]内的表达式进行计算，得到 1 个函数值向量 y，接下来对 y 的数值进行修正，修正的格式可写为：

```
y(条件语句*) = f(x(条件语句*)); % 统一格式修改分段函数取值
```

在上面的格式中，"条件语句*" 用来确定 x 的范围，右侧是关于 x 的函数表达式。等式两侧 "条件语句*" 的内容要一致。可以比较程序中第 6 行和第 7 行的内容以理解上面介绍的统一格式。

调用函数文件 piecewise2.m，利用下面的语句可以得到分段函数 $f(x)$的曲线图。

```
x = linspace(0,4);     % 自变量取样点集
y = piecewise2(x);     % 计算相应函数值
plot(x,y);             % 绘制曲线
hold on;
plot(1*ones(1,2),ylim,'r:');  % 画区间间隔线
plot(2*ones(1,2),ylim,'r:');  % 画区间间隔线
set(gca,'fontsize',12);
xlabel('\itx','fontname','times new roman','Fontsize',14);
```

所得曲线如图 6.7 所示，这个分段函数在区间端点处是连续的，有时也存在区间端点不连续的情况，利用上面的思路同样可以计算分段函数各点的函数值。

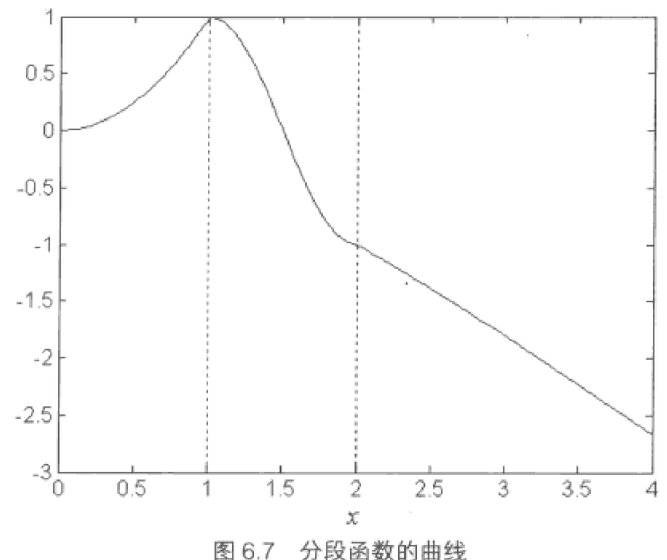


图 6.7 分段函数的曲线

当分段函数表达式比较简单的时候，可以使用 inline 函数来定义分段函数。使用如表 6.5 所示的语句（该部分内容保存在光盘中的 piecewise2p.m 文件中）可以实现公式（6-1）中的分段函数。

表 6.5 定义分段函数的语句

行号	程序内容	
1	s1 = 'x.^2.*(x>=0&x<1)';	% 第 1 区间内的表达式
2	s2 = 'cos([x-1]*pi).*(x>=1&x<2)';	% 第 2 区间内的表达式
3	s3 = '(-1)*x.^2./[x+2].*(x>=2)';	% 第 3 区间内的表达式
4	fx = inline([s1,'+',s2,'+',s3]);	% 联合起来成为分段函数
5	y = fx(linspace(0,4));	% 计算函数值
6	plot(linspace(0,4),y);	% 绘图

在上面的程序中利用了 inline 函数定义这个分段函数，使得该程序可以在一个脚本文件中完成计算和绘图工作，所得结果与图 6.6 一样。这里第 1~3 行是定义各段区间上的函数表达式，除了表达式之外还有一个区间判断语句，即“(x>=0&x<1)”、“(x>=1&x<2)”和“(x>=2)”，它们将返回 0 和 1 构成的一个逻辑性数组，还可以和前面的函数值进行点乘计算，进行一个采样操作。通过第 4 行把 3 段函数表达式联合在一起，直接相加就得到了公式（6-1）所定义的函数。

6.2.5 子函数和私有函数

在函数文件中允许定义多个子函数。函数文件中第一个 function 标识的为主函数，其他 function 标识的为子函数。子函数只能由这个函数文件的主函数调用，或者由它前面的子函数调用而不能由其他 M 文件调用。每一个子函数和函数文件的定义类似，在定义的时候需要根据先后调用顺序安排子函数的顺序。表 6.6 是子函数例子，这里仅附上程序的框架，完整程序请见光盘中 main_sub.m 文件。

表 6.6　一个子函数的例子

行号	程序内容
1	function y = main_sub(n);
2	if mod(n,2) == 1;
3	y = magic_odd(n);　% 奇数时执行子函数进行计算
4	elseif mod(n,2) == 0;
5	error('n must be an odd number');　% 偶数时返回错误提示
6	end
7	function M = magic_odd(n);
8	% 该函数部分内容略去
9	[x,y] = modfun(x,y,n);　% 调用 modfun 子函数
10	function [x,y] = modfun(x,y,n);
11	x = mod(x-1,n)+1;　% 计算 x 对 n 的模除，使结果在[1, n]范围内
12	y = mod(y-1,n)+1;　% 计算 y 对 n 的模除，使结果在[1, n]范围内

这是一个用来计算奇数阶幻方矩阵的程序，其中 magic_odd 和 modfun 都是这个程序中的子函数，子函数 modfun 由 magic_odd 调用，因此子函数 modfun 需要放置在子函数 magic_odd 的后面。

私有函数是 MATLAB 函数文件的一种形式。它唯一的特征是只能够在一个特定的限定函数群体中可见。如果要约束函数的访问范围，或者当选择不让外面看到所执行的是哪个函数的时候，可以用到私有函数。私有函数存放在以专有名称 private 命名的子目录下，它们只能由其父目录中的函数文件调用。比如在光盘中\ch06\private 目录下的 rotate_shift 文件，只要把该文件复制到硬盘上之后，在命令窗中输入：

```
>> [R,S,K]=rotate_shift(1,2,3,4,5,6,7,8)
```

将会出现下面的提示：

```
??? Undefined function or variable 'rotate_shift'.
```

也就是说在其父目录中的脚本文件也不能调用 private 目录下的函数文件。然而定义如下函数：

```
function private_test; % 私有函数的测试
 [R,S,K]=rotate_shift(1,2,3,4,5,6,7,8)
```

执行 private_test 函数就可以调用 rotate_shift 函数了。

因为私有函数在其父目录以外调用，是"隐身"的，因此可以与其他目录下函数文件的名字相同。这一点在创建用户特定函数新版本的时候同时在另外目录中保存原来版本的函数是非常有用的。在 MATLAB 中先查询私有函数，再寻找其他位置的 M 文件。此外，help，lookfor 和 which 等命令在 private 目录以外，不能提供关于私有函数的任何帮助信息。

6.3　函数文件与脚本文件的比较

函数文件与脚本文件的主要区别在于：函数文件一般都要带参数，都要返回结果（也有一些函数文件不会带参数和返回结果的），而且函数文件还要定义函数名；脚本文件没有参数和返回结果，也不在程序的开头定义函数名，通过生成和访问工作空间中的变量可以与外界和其他函数交换数

据。另外，脚本文件的变量在文件执行结束后仍然会保存在内存中不丢失；函数文件的变量仅在函数执行期间有效，当函数执行完毕，它所定义的所有过程变量都会被清除（只有输出的部分变量可以在内存中查看到）。使用函数 whos 不能看到这些过程变量，在函数运行期间使用函数 whos 可以看到函数内部定义的变量以及输入参数。函数中定义的变量可以与工作空间中的变量重名，而不会影响到工作空间中原来存储的变量。

对于程序的调试工作而言，脚本文件要比函数文件方便些。因为脚本文件的过程变量的 size 可以通过 workspace 来查看，一些数组的“超维”调用问题以及不合法的矩阵乘法可以直接从 work space 窗口发现。如果注释掉函数文件第一行带“function”的语句，同时对所有输入变量赋值，那么函数文件就可以转化为脚本文件。对于初学者，建议使用脚本文件进行程序调试工作。当然在函数文件中，去掉分号也可以显示过程变量的取值，可以查看程序计算的数值和预期的是否相符。

编好的函数文件可以用于日后在其他问题中反复进行调用，如果这样的函数文件比较多了，使用者就有了自己的“工具箱”，把这些函数放置在 MATLAB 的搜索路径中就可以像 MATLAB 自带函数一样放心使用了。频繁使用的程序最好写为函数文件的形式，这样可以方便管理，同时能够加快程序执行速度。

广大研究者经常根据任务需要，解决不同类型的科学和工程问题，对于程序的分类管理和注释是非常重要的。对于简短的函数文件，程序中添加部分文字可能就能满足注释的要求。对于大型程序，建议使用者另附详细文档（如 Word 文档）进行补充性注释，其中可以增加流程图、公式等无法进行文本输入的内容，详细的思路记载也是需要的。

6.4 数据文件

数据文件的读入和写入是不同软件之间交流的有效途径之一。目前一些流行软件的高级版本已经支持 MATLAB 输出的数据文件类型（即.mat 文件）。本节将考虑一些数据文件格式的读入和写入问题，如.dat，.txt，.xls 等。通过输入 help iofun 可以得到数据、图片、声音和视频等文件读写相关函数的列表。本节将详细介绍数据读写函数的使用方法。

6.4.1 常用的数据文件读入函数

常用的数据文件读入函数有 load，importdata，dlmread，wk1read，xlsread，fopen，fgetl，fscanf 等。下面具体介绍这些函数的使用。

6.4.1.1 数据读入函数

load 函数是最常用的数据读入函数，其调用格式为：

```
load filename                    % 格式 1
load filename x y z ...          % 格式 2
data = load('filename');         % 格式 3
```

参数说明：函数 load 一般可以读入.dat，.txt，.mat，.xls 等数据文件。参数 filename 可以是数据文件名，也可以是含路径的文件名（这里文件名需要写上扩展名），比如 D:\data1.txt。其中格式 1 将读入数据文件中的所有数据信息，格式 2 可以指定读入数据文件中变量对应的数据，格式 3 可以读入路径名或者文件名中含有空格符的数据文件，其作用和格式 1 相同。此外格式 3 在处理大

量数据时会很方便，把文件名依次循环输入即可，比如需要处理 Su1.dat，Su2.dat，Su.3.dat，等等，可以使用如下语句：

```
for k=1:200;
    D = load(['Su',num2str(k),'.dat']);     %进行数据处理
end
```

6.4.1.2 导入数据文件

函数 importdata 可以读入数据文件、音频及图片文件的数据变量信息，其调用格式如下：

```
s = importdata('filename');
```

参数说明：在函数 importdata 使用中要求文件名（含扩展名）以字符串形式输入。当数据不在当前目录时，需要把完整目录写上，如 E:\Liu\Data1.txt。当数据文件中含有非数据的中英文头文件或者其他说明时，s 是细胞矩阵结构，为了得到目标数据还需要做进一步的处理。对于一些特殊软件生成的数据文件，可以尝试使用函数 importdata 读入数据。比如下面的语句可以读入 T0624.txt 文件中的数据。

```
s = importdata('D:\MATLAB\book1\ch06\T0624.txt')
```

6.4.1.3 读取 ASCII 文件到矩阵

函数 dlmread 可以读取带有分隔符的 ASCII 文件到矩阵函数，其调用格式为：

```
s = dlmread('filename');
```

参数说明：比如在一个记事本文件中输入"7.1;8.2;6.3;6.2;5.3,9.1"，并把这个文件命名为 DLM.txt，利用语句 A=dlmread('DLM.txt')可以得到：

```
A = 7.1000    8.2000    6.3000    6.2000    5.3000    9.1000
```

6.4.1.4 读入 Lotus1-2-3 的数据文件

函数 wk1read 可以用来读入 Lotus1-2-3 电子数据表文件的数据（这是一种类似 Excel 表格的文件），这个函数的调用方法和 dlmread 函数类似，这里不再赘述。比如下面的语句可以把 WK1_1.wk1 文件中的数据读入 workspace。

```
s = wk1read('D:\MATLAB\book1\ch06\WK1_1.wk1')
```

6.4.1.5 读入 Excel 数据文件

函数 xlsread 可以读入 Excel 数据文件，其调用格式如下：

```
A = xlsread('filename');                      % 格式 1
A = xlsread('filename',Sheet);                % 格式 2
A = xlsread('filename',Sheet,range);          % 格式 3
```

参数说明：格式 1 将读入 Excel 数据文件的第一个 Sheet 里面的内容。格式 2 可以读入指定 Sheet 内的数据，其中 Sheet 可以是目标 Sheet 的名称（此时需要输入相应的字符串），也可以是 Sheet 的序号。格式 3 可以在 Sheet 中指定一个子区域进行数据读取，其中 range 是子区域左上和右下网格的坐标，如 A2:J5 或者 a2:j5。下面是调用 xlsread 的一个例子。

```
s1 = xlsread('D:\MATLAB\book1\ch06\XLS1.xls',1,'A2:J5')
```

它可以得到指定范围内的数据。

6.4.1.6 fopen 读取特殊数据

某些特殊数据文件使用前面的函数可能无法读取，可以考虑使用 fopen 和 fgetl。首先介绍一下 fopen 函数的使用方法，它的调用格式为：

```
fid = fopen(filename, permission);
[fid, message] = fopen(filename, permission);
[fid, message] = fopen(filename,permission, format);
[filename, permission, format] = fopen(fid);
```

参数说明：fid 是要读入文件的标识，它是一个整数。filename 是文件名，包含扩展名。permission 用来定义打开数据文件的模式。format 用于指定文件中数据的格式。若文件读入成功，fid 是一个正整数，同时 message 是一个空字符串；若文件打开失败，fid 是一个负数，同时 message 返回出错信息。利用参数 message 在数据的批量读入时很方便，程序不会因出错而停止，根据 message 的值可以制定不同的处理方式。表 6.7 给出 permission 可输入的字符串及意义。

表 6.7 参数 permission 的可选字符串及意义

字符串	意义
r	只读方式打开文件
r+	可读写方式打开文件
w	删除文件中的内容，并以只写格式打开
w+	删除文件中的内容，并以只读格式打开
a	只写方式打开文件，并可把新的内容只写增加到文件尾
a+	只写方式打开文件，并可把新的内容可读写增加到文件尾
W	不进行自动刷新写入数据
A	不进行自动刷新增加数据

如果要以文本文件的方式打开数据文件，我们应该增加字母 “t” 到 permission 项中（如 wt 和 wt+）。当以二进制模式打开数据文件的时候，需要在 permission 里面增加字母 “b”（如 wb）。特别地，文本文件和二进制文件在 UNIX 操作系统中是没有区别的，因此在此系统上，字母 r 和 b 是不需要加以区分的。

在调用函数 fopen 的时候，参数 format 记录字符串数据存储在文件中的格式。在两台计算机中传递互相矛盾的数据格式时，这个字符串 format 是必须用到的。一些可能格式的控制字符为 n，l，b，d，g，c，a，s，可以从中选择使用。

6.4.1.7 函数 frewind 读取文件

通过函数 frewind 来进行文件读取的时候，可以把指针放到文件头。下面是一个数据读入的例子：

```
function y=readdata(str); % str 是文件名
fid=fopen(str); % 打开数据文件
n=1;     % 初始化累加参数 n
y=[];    % 初始化输出量为空矩阵
while n>0;
    tline = fgetl(fid); % 读取当前行
    if sum(findstr(tline,'<<'))<0.5&n>6; % 检查是否满足条件
```

```
      y=[y;str2num(tline)]; % 读入数据
   elseif sum(findstr(tline,'<<'))>0.5
      break; % 退出循环
   end
   n=n+1; % 累加
end
fclose(fid); % 关闭数据文件
```

上述程序可以实现从第 6 行开始读数据。有些特殊软件输出的数据一般能用记事本打开，这里可以看到其中的头文件信息，也可以读出从哪一行开始是数据部分。这里通过检查最后一行出现了特征字符“<<”，即将停止程序。

6.4.1.8 读取格式化数据

函数 fscanf 可以实现从一个数据文件中按用户自定义的格式来读取格式化数据，它的调用格式如下：

```
array = fscanf(fid, format);
[array, count] = fscanf(fid, format, size);
```

参数说明： fid 是利用 fopen 函数得到的数据文件标识。参数 format 是可供用户控制如何读取数据格式的字符串，表 6.8 给出了一个详细说明。

表 6.8 参数 format 可能取值及含义

字符串	含义
%a	读取一个字符串，其可被空格符或其他类似于换行符的特殊符号分开
%c	读取一个字符，可以是任意类型的字符，如空格，换行符等
%d	读取一个小数（忽略空格）
%e	读取一个小数（忽略空格）
%f	读取一个小数（忽略空格）
%g	读取一个小数（忽略空格）
%i	读取一个有符号数据（忽略空格）
%o	读取一个八进制数据
%s	读取一个字符串（忽略空格）
%u	读取十进制数据
%x	读取十六进制数据



在表 6.8 中的符号“%”后面可以加入数字 N 来控制读取的位数。参数 size 可以用来指定从文件读取数据的数目。这个函数有以下 3 种取值类型：（1）n 准确地从数据文件中读取 n 个值，执行函数 fscanf 之后，输出 array 是一个包含 n 个值的列向量；（2）inf 读取数据文件中所有值，在执行函数 fscanf 之后，输出 array 是一个列向量，它包含了数据文件中所有值；（3）[m,n] 从数据文件中精确地读取 m×n 个值，此时输出 array 是一个 m×n 的数组。若数据文件中的数据与格式转换指定字符不相匹配，那么函数 fscanf 将会中止执行。

下面举一个例子讨论函数 fscanf 的用法，比如在记事本程序中写入下面的内容：

```
1 2 3 4
5 6 2 3
```

```
2 2 4 5
```

将上面的内容保存为 ttz.txt 文件。利用函数 fscanf 来读入这个数据文件：

```
fid = fopen('ttz.txt');        % 打开 ttz.txt 文件
A = fscanf(fid,'%u',inf);     % 读入数据
B = A'                        % 显示 A 的数据内容
A = reshape(A,[4,3]);         % 重新排列数据顺序
A = A'                        % 显示整理后的数据
fclose(fid);                  % 关闭 fid
```

上述程序执行后结果如下：

```
B =    1    2    3    4    5    6    2    3    2    2    4    5
A =
     1    2    3    4
     5    6    2    3
     2    2    4    5
```

从 B 的数据内容可以看出函数 fscanf 读入数据是逐行读入的，将下面的程序与上面的程序进行对比：

```
fid = fopen('ttz.txt');        % 打开 ttz.txt 文件
A = fscanf(fid,'%u',[3,4])     % 读入数据
fclose(fid);                   % 关闭 fid
```

这段程序的输出如下：

```
A =
     1    4    2    2
     2    5    3    4
     3    6    2    5
```

可见，此时 A 的顺序和 ttz.txt 文件中的顺序不一致，因此用户在调用这个函数的时候需要注意这个问题，并合理利用 reshape 函数来调整顺序。

6.4.2 常用数据的写入函数

数据的写入可以使用 save，fwrite，fprintf，csvwrite，wk1write，xlswrite，cdfwrite 和 kmlwrite，下面具体介绍这几个函数的使用。

6.4.2.1 函数 save 保存数据

函数 save 可以把 MATLAB 计算生成的数据变量保存为.mat，.dat，.txt 等常见数据文件格式，其调用格式如下：

```
save filename          % 格式 1
save filename x        % 格式 2
save filename x y z    % 格式 3
save filename A*       % 格式 4
```

参数说明： 这里 filename 是文件名，如果不写扩展名，MATLAB 将把数据文件的扩展名默认为.mat。格式 1 将把当前 workspace 里面的所有变量保存到数据文件，在数据文件中组成一个结

构体。格式 2 可以保存指定的数据变量。格式 3 可以指定多个变量进行数据存储。格式 4 可以使用通配符 "*"，其可以写为不同形式，如 "A*"，"*B" 和 "*B*" 等，在需要保存多个具有共同字符的数据文件时，利用通配符是很方便的。如下面的语句：

```
A = rand(4,3)
save randdata A
```

可以把变量 A 存入 randdata.mat 文件中。

6.4.2.2 二进制格式储存数据

函数 fwrite 可以实现以二进制格式储存数据，这个函数的使用需要结合 fopen 和 fclose 来实现。函数 fwrite 的调用格式如下：

```
count = fwrite(fid, array, precision);
count = fwrite(fid, array, precision, skip);
```

参数说明： fid 是使用函数 fopen 打开的数据文件标识，array 是欲写入的数据矩阵。count 是写入数据文件变量的个数。参数 precision 用来指定写入数据的格式，为了便于数据文件在不同计算机或者不同软件之间传递，指定写入数据的格式是非常重要的，表 6.9 给出了相应的字符及意义。

表 6.9 参数 precision 可能取值及含义

字符串	C/Fortran 形式	含义
char	char*1	6 位字符型数据
schar	signed char	8 位有符号字符型数据
uchar	unsigned char	8 位无符号字符型数据
int8	integer*1	8 位整型数据
int16	integer*2	16 位整型数据
int32	integer*4	32 位整型数据
int64	integer*8	64 位整型数据
uint8	integer*1	8 位无符号整型数据
uint16	integer*2	16 位无符号整型数据
uint32	integer*4	32 位无符号整型数据
uint64	integer*8	64 位无符号整型数据
float32	real*4	32 位浮点型数据
float64	real*8	64 位浮点型数据
bitN		N 位有符号整型数据（1≤N≤64）
ubitN		N 位无符号整型数据（1≤N≤64）



在表 6.9 中对于数据文件的大小，除了 "bitN" 和 "ubitN" 以位为单位，其他格式的数据都是以字节为单位的。使用参数 skip 可以指定在每一次写入数据之前跳过的约定的字节数。在替换有固定长度的字符时，使用这个参数非常方便。特别地，当 precision 等于 bitN 和 ubitN 时，skip 表示跳过 skip 位。

下面结合一个例子说明这个函数的使用。

```
fid = fopen('magic5.txt','w+b');          % 打开数据文件进行读写
fwrite(fid,magic(5),'int8');              % 写入 int8 型数据
```

```
fclose(fid);
```

对于函数 fwrite 得到的数据文件一般情况下不能使用 load 和 importdata 读入，而需要利用函数 fread 来读取其中的数据，如读入上面语句生成的数据文件 magic5.txt，可以使用如下语句：

```
[fid,message] = fopen('magic5.txt');
Cm=fread(fid,[5,5],'int8')
```

程序输出如下：

```
Cm =
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
11    18    25     2     9
```

在读取数据的时候需要明确数据文件中的数据格式，否则将会出现错误。比如执行下面的语句：

```
Cm=fread(fid,[5,5],'int16')
```

将会输出下面的结果：

```
Cm =
        5905          1793          4111
        2564          4877           790
        6155          2073             0
        1541          5134             0
        4620           533             0
```

6.4.2.3　把特定格式化数据写入文件

函数 fprint 可以根据用户自定义格式把特定格式化数据写入一个文件中，它与 fscanf 函数是一对读写函数，其调用格式为：

```
count = fprintf(fid, format, val1, val2, ...);
fprintf(fid, format, val1, val2, ...);
```

参数说明：参数 fid 是数据文件标识。如果 fid 缺省，数据将写入到命令窗中。参数 format 是用来控制数据写入格式的字符串，其参数意义与表 6.8 等同。在参数 format 中可以用字符串指定队列长度、小数部分位数、整数部分位数和输出格式等，它包括文字、数字字符（位于符号“%”和格式字母之间）和字母符号（用于指定输出数据显示的精确格式）。一个典型的数据输出格式控制字符串表示为“%8.6e”，其中“%”是控制符开始标志，“8”表示整数部分位数，“6”表示小数部分位数，e 是格式描述。这里的字符“%”总是标志着格式化字符串的开始，在其之后的字符串包括整数部分位数控制数字、精度指定符和转换指定符。

在函数 fprintf 中，可以利用下面几个字符进行格式控制：减号“-”用来控制数据左对齐，如果这个控制符缺省则认为是右对齐；加号“+”表示写入数据时带有正负号；“0”表示如果数据的位数不足，用 0 填充前面的空位。比如，可以用 fprintf(fid,' %-8.6e',A)来控制对齐方式。

如表 6.10 所示给出了一些格式字符串的转义字符及相应含义，它们可以以字符的形式输入到 format 项中。

表 6.10 格式字符串的转义字符

字符串	含义	示例
\n	换行符	fprintf('%d\n',123)
\t	水平制表符	fprintf('%f\t',234)
\b	退后一格	fprintf('%f\b',234)
\r	回车符，使光标移到下一行最前面	fprintf('%d\r',234)
\f	跳页符号	fprintf('%d\f',234)
\\	输入一个反斜杠	fprintf('%e\\',234)
... or \"	输入一个省略号或单一引号	fprintf('%e...',234),fprintf('%e\"',234)
%%	输入一个百分号（%）	fprintf('%d%%',234)

下面的一段程序可以把一段数据写到一个记事本文档中。

```
x = 1:128;                    % 生成数据
filename = 'vector1.txt';       % 文件名
fid = fopen(filename,'wt');    % 打开文件
for p = 1:length(x);
   fprintf(fid,'%d',x(p));    % 写入数据
   fprintf(fid,'%c\n',' ');    % 输入换行符
end
fclose(fid);                   % 关闭文件
```

6.4.2.4 把数据写入逗号分隔值文件

函数 csvwrite 可以把数据写入逗号分隔值文件（comma separated value，缩写为 csv），该文件类型可以使用 Excel 软件打开，该函数的调用格式如下：

```
csvwrite(filename,m);
csvwrite(filename,m,r,c);
```

参数说明： filename 是 csv 文件的名称，含扩展名。m 是数据矩阵。r 和 c 分别表示 csv 文件中左上角数据离开上边和左边的行数和列数，它们的默认值都是 0。比如利用下面的语句我们可以得到一个 csv 文件。

```
csvwrite('csv1.csv',rand(4,3),1,2)
```

6.4.2.5 数据写入 wk1 文件

函数 wk1write 可以把数据写入 wk1 文件中，其调用格式和参数意义与函数 csvwrite 相同。比如语句 wk1write('WK1_1.wk1',magic(4))可以得到 WK1.wk1 文件。

6.4.2.6 把数据写入 xls 文件

函数 xlswrite 可以把数据写入 xls 文件中，其完整调用格式如下：

```
[success,message]= xlswrite(filename,array,sheet,range)
```

参数说明： success 返回写入是否成功，message 返回可能的出错信息。filename 是 xls 文件名称。array 是数据矩阵。sheet 表示 sheet 的序号。range 用来约束数据写入的范围。在实际调用中，filename 和 array 是必须输入的，其他参数可以缺省。这个函数在早期版本中不存在，用户可以在网上搜索并利用这个文件即可。下面是一个调用 xlswrite 的例子：

```
xlswrite('majic.xls',magic(4),2,'b2:e5')
```

6.4.2.7 把数据写入频道定义格式

函数 cdfwrite 可以把数据写入频道定义格式（Channel Definition Format，缩写为 CDF），其调用格式如下：

```
cdfwrite(filename, variablelist);
```

参数说明：filename 是 cdf 文件名称。variablelist 是欲写入的数据。更多详细属性设定，用户可以参考该函数的帮助信息。此外，cdf 文件的读入函数是 cdfread。下面是一个调用 cdfwrite 函数生成 cdf 文件的例子：

```
cdfwrite('CDF1.cdf', {'HIT', 1920:1922});
```

6.4.2.8 把数据写入 kml 文件

函数 kmlwrite 可以把数据写入 kml 文件。读者可详细阅读该函数的帮助信息来了解该函数的使用。下面是生成 kml 文件的例子：

```
address = {'Harbin', ...
         'China', ...
         'Heilongjiang'};
filename = 'mesage.kml';
kmlwrite(filename, address, 'Name', address);
```

6.5 图片文件

在 MATLAB 中函数 imread 和 imwrite 可以读写图片文件。此外函数 saveas 和 print 可以把 figure 上的曲线或者图像保存为图片格式。下面详细介绍这 4 个函数的使用。

6.5.1 读入多种格式的图片文件

函数 imread 可以读入多种格式的图片文件，具体支持的格式如表 6.11 所示。

表 6.11 函数 imread 支持的图片格式

格式	说明
JPG	支持所有这种格式的图片
TIFF	1，4，8，24 位非压缩和 packbit 压缩图像，16 位灰度和索引图像，1 位 CCITT 压缩和 48 位 RGB 图像
GIF	1 位和 8 位 GIF 图像
BMP	1，4，8，16，24，32 位非压缩图像，4 位和 8 位行程长度编码（run-length encoded，缩写为 RLE）图像
PNG	任意 PNG 图像，包括 1，2，4，8，16 位灰度级图像，8 位和 16 位索引图像，24 位和 48 位 RGB 图像
HDF	8 位和 24 位光栅图像数据库
PCX	1，8，24 位 PCX 图像
XWD	1 位和 8 位 Zpixmaps，XYbitmaps，1 位 XYPixmaps
ICO	1，4，8 位非压缩 ICO 图像
CUR	1，4，8 位非压缩 CUR 图像

（续表）

格式	说明
RAS	任意 RAS 图像，包括 1 位位图，8 位索引图像，24 位真彩色图像，32 位 alpha 通道真彩色图像
PBM	任意 1 位 PBM 图像
PGM	任意标准 PGM 图像
PPM	任意标准 PPM 图像



尽管 MATLAB 支持的图片格式很多，但还有一些图片格式不能直接读入。为此在使用图片文件的时候，可以考虑使用第三方软件来转换格式。

函数 imread 的调用格式如下：

```
A = imread(filename,FMT);
A = imread(filename);
A = imread(filename, IDX);
[A, map, alpha] = imread(filename);
```

参数说明： A 是图片文件对应的像素矩阵。filename 是图片文件名，它可以包含或者不包含扩展名，当 filename 中不包含扩展名的时候需要在参数 FMT 中写上扩展名。参数 IDX 用于读入.tiff，.ico 和.gif 等图片格式时，对多帧图像中某一帧的读入。map 是像素映像信息。alpha 是 alpha 通道信息。

下面是几个读入图片的例子。

```
A = imread('cameraman','tif');    % 读入摄像师照片
A = imread('rice.tif');        % 读入米粒图
[B, map] = imread('pout.tif');  % 读入一个小孩的照片
[B, map,alpha] = imread('magic20.png');  % magic20.png 是光盘中的一个随机图像
[B, map] = imread('mri.tif',2);  % 读入多帧图像中的一帧
```

6.5.2 把数据写到一个图片文件

函数 imwrite 可以把数据写到一个图片文件中，其调用格式如下：

```
imwrite(A, filename);
imwrite(A, filename, fmt);
imwrite(A, map, filename, fmt);
imwrite(A, filename, fmt, param1, val1, param2, val2,...);
```

参数说明： 参数 A 是欲写入的数据，其要求是 uint8 或者 uint16 的数据格式，如果是 double 型数据，需要利用 uint8(A-1)进行转化。map 是一个有效的 MATLAB 支持的 colormap 格式。filename 是图片文件名。fmt 是扩展名。对于不同图片格式，用户可以对图片的具体参数进行设定，请参阅 imwrite 的帮助信息。

对于 dcm 格式的图片可以使用函数 dicomread 和 dicomwrite 进行该图片格式的读写操作。具体用法和函数 imread 和 imwrite 类似，这里不再赘述。

函数 imfinfo 可以用来查看图片文件的信息，返回结果包含了详细的文件信息。为了方便使用 MATLAB 自带的图片信息，本书在光盘的 image_list.doc 文件中列举了图片处理工具箱提供的图片

名称及相关信息。

6.5.3 把矩阵保存为图片文件

函数 imwrite 把一个矩阵或者 M×N×3 的数组保存为图片文件，对于 Figure 上的曲线和图像的保存还可以使用函数 saveas 和 print 函数。函数 saveas 的调用格式如下：

```
saveas(h, filename, format);
```

参数说明： h 是 figure 的句柄，当前图形窗口的默认句柄是 gcf。filename 是文件名。format 是扩展名。当 format 缺省且 filename 里面也未写扩展名时，则保存的文件扩展名为 fig。在实际应用中，建议大家在保存的时候选择两种格式：fig 和需要的文件格式。fig 格式的文件可以继续读入 MATLAB 进行编辑，起到备份的作用。

6.5.4 打印当前图形文件

函数 print 可以把当前的（或者指定的）图形窗口上面的内容打印到.jpg，.eps 和.tif 等格式的图形文件中，其调用格式为：

```
print -device -options
print(h, '-device', filename)
```

参数说明： device 用来设置打印方案，如黑白、彩色等格式。options 可以用来设置一些参数，如 dpi 等。filename 是文件名。

下面是相关的 4 个例子。

```
print('-dpsc','-r600','hhh.eps')      % 打印当前图形窗口成 eps 文件，并设置 dpi 数值为 600
print -dpsc -r200 hhh.eps             % 打印当前图形窗口成 eps 文件，并设置 dpi 数值为 200
print(gcf, '-djpeg', 'foo')           % 打印当前图形窗口成 jpg 文件
print -depsc -tiff -r300 matilda      % 打印当前图形窗口为 tiff 文件，并设置 dpi 为 300
```

-r300 和-r600 分别用于设置 dpi。在一些学术文章中往往需要设置图形采用一定的 dpi 数值，可以使用上面的语句来设定。在 MATLAB 的高级版本 figure 属性中可以设定 dpi 等属性，也可以方便地利用语句来实现，这样便于操作，能放更多精力在程序的调试上。

6.6 视频和音频文件

MATLAB 不太支持视频文件，对于 AVI 文件可以使用函数 aviread 来读入所有的帧或者特定帧。函数 aviread 的调用格式为：

```
mov = aviread(filename);
mov = aviread(filename,index);
```

参数说明： mov 是返回的.avi 文件信息，其中包含帧画面图像信息。filename 是.avi 文件名称。index 可以用于指定某一帧或者多帧。

下面是读入.avi 文件的例子。

```
A=aviread('aviF.avi',6);  % 读入aviF.avi文件的第6帧
```

这里 A 是一个结构体，包括 cdata 和 colrmap 两部分内容，其中 cdata 是像素数据信息，colormap 是颜色映像信息。

函数 avifile 可以用来记录一个 avifile 文件，其调用格式如下：

```
aviobj = avifile(filename);                                          % 格式 1
aviobj = avifile(filename, 'propertyname', value, 'propertyname', value, ...);  %
格式 2
```

参数说明：aviobj 是生成.avi 文件的标识，是一个结构体数据，其作用相当于句柄。filename 是.avi 文件名。此外，用户还可以设定.avi 文件的一些参数，包括 FPS 指定每秒的帧数，默认值是 15；COMPRESSION 设定压缩方式，可以选择的参数有 Indeo3，Indeo5，Cinepak，MSVC，RLE 或者 None，WINDOWS 系统默认值是 Indeo3，UNIX 系统默认值是 None；QUALITY 用于控制压缩程度（当无压缩时，这个参数无效），QUALITY 越大，.avi 文件的质量越好，该参数默认值为 75；KEYFRAME 指每秒特定帧数目，其默认值是 2，此参数是针对于支持临时压缩的情况；COLORMAP 是颜色映像表，其大小是 N×3，其中 N 不能大于 256（当存在压缩时，该值是 236）；NAME 是视频文件描述名称，它不能多于 64 个字符。除了像格式 2 那样定义具体参数的数据外，也可以按下面的方式定义：

```
aviobj = avifile(filename);
aviobj.Quality = 99;
```

下面是用 avifile 函数记录一个同心圆环变化的过程。

```
axis square;                          % 设置坐标轴为方形
set(gcf,'DoubleBuffer','on');            % 使图像变化是图形窗口不发生抖动
set(gca,'xlim',[-80 80],'ylim',[-80 80],...
    'NextPlot','replace','Visible','off');        % 设置坐标轴属性
mov = avifile('example.avi','Quality',100);     % 生成.avi 文件及定义属性
x = -pi:.1:pi;              % 采样数据
radius = [0:length(x)];  % 半径采样数据
for i=1:length(x)
    h = patch(sin(x)*radius(i),cos(x)*radius(i),[abs(cos(x(i))) 0 0]);   % 利用数
据绘图
    set(h,'EraseMode','xor');    % 设置擦除方式为异或
    F = getframe(gca);         % 获取图形窗口像素信息
    mov = addframe(mov,F);  % 把F填加到mov中作为一帧
end
    mov = close(mov);  % 关闭mov
```

此时可以删除生成的.avi 文件，并不需要关闭 MATLAB 才能删。

特别需要指出的是，在利用 avifile 记录图形窗口变化的时候，被记录的图形窗口不能被其他界面遮挡，即使系统锁定、注销也不可以。若图形窗口被遮挡，记录的将是被遮挡后的结果。因此在长时间记录的时候，用户需要取消系统的屏幕保护程序。

函数 auread 可以读入.au 音频文件，具体调用格式如下：

```
y = auread(aufile);
[y,fs,bits] = auread(aufile);
y = auread(aufile,n);
```

```
[y,fs,bits] = auread(aufile,[n1 n2]);
siz = auread(aufile,'size');
```

参数说明：y 是返回的音频文件的采样数据。fs 是采样率。bits 是每个采样点的 bit 数。aufile 是.au 文件名。n 表示返回前 n 个通道的采样数据。[n1, n2]表示返回通道 n1 和 n2 之间的采样数据。当使用参数 size 的时候，将返回采样和通道信息，即 SIZ=[samples channels]。比如利用 auread 函数读入 aufile.au 可以用下面的语句:

```
auread('aufile.au')
```

函数 auwrite 可以把数据写入.au 音频文件中，其调用格式如下：

```
auwrite(y, aufile);
auwrite(y, fs, aufile);
auwrite(y, fs, bits, aufile);
auwrite(y, fs, bits, method, aufile)
```

参数说明：除了 method 以外，其他参数与函数 auread 中的参数意义相同，参数 method 可以选择的参数有 mu 和 linear，其中 mu 是默认值，选择 mu 参数时，y 需要的是 8 位数据。

生成.au 文件的例子如下：

```
auwrite(rand(1,20000),'aufile.au')
```

这里把一个随机序列存入.au 文件。

函数 wavread 和 wavwrite 可以读写.wav 音频文件，其用法和函数 auread 和 auwrite 基本一样，可以参考它们的帮助信息，这里不再赘述。

6.7 文件批处理结构

如果在编程的时候能建立文件批量处理的程序，将是一件非常有意义和有意思的事情。为了方便建立批处理程序，下面介绍基本的 MATLAB 文件处理函数，具体如表 6.12 所示。

表 6.12 基本文件管理函数

函数名	说明	函数名	说明
cd	改变工作路径	ls	作用同 dir
copyfile	复制文件	matlabroot	返回 MATLAB 安装根目录
delete	删除文件	mkdir	建立新路径
diary	保存会话	open	打开 M 文件
dir	列出当前路径下的所有文件	pwd	返回当前路径
edit	编辑 M 文件	rmdir	删除路径
fileparts	返回文件名的信息	tempdir	返回系统临时路径
fullfile	建立文件	type	把 M 文件内容输入命令窗
inmem	返回内存中的函数	what	列出 MATLAB 文件

下面依次介绍这些函数的使用。

6.7.1 改变 MATLAB 的当前路径

函数 cd 可以改变 MATLAB 的当前路径，其调用格式为：

```
cd 或者 f = cd            % 格式 1
cd directory              % 格式 2
cd('directory')           % 格式 3
cd ..                     % 格式 4
```

参数说明： 格式 1 中将返回当前路径，f 是记录当前路径的一个字符串，此时函数 cd 的作用相当于函数 pwd。格式 2 中，将把当前路径转移到路径 directory。如果在路径名中含有空格，比如想进入下面的路径：

```
D:\Mfiles\Ahit\book1\ch06\Ax Bx
```

那么使用“cd D:\Mfiles\Ahit\book1\ch06\Ax Bx”将会出现如下错误提示：

```
??? Error using ==> cd
Too many input arguments.
```

在这样的情况下，可以调用格式 3，即 cd('D:\Mfiles\Ahit\book1\ch06\Ax Bx')，这样就可以正确地进入这个带有空格的路径了。格式 4 将返回上一级目录，它可以由格式 2 和格式 3 代替。MATLAB 提供了函数 matlabroot 来获得 MATLAB 软件安装的根目录，其调用格式如下：

```
matlabroot 或者 s = matlabroot
```

参数说明： s 是根目录的字符串信息。

6.7.2 复制文件

函数 copyfile 可以复制文件，其调用格式如下：

```
copyfile ('source', 'dest');
copyfile ('source', 'dest', 'writable');
status = copyfile ('source', 'dest');
[status, msg] = copyfile ('source', 'dest');
```

函数 copyfile 把指定源文件目录中的文件 source 复制到指定的路径 dest 中。如果源文件位于当前路径下，那么 source 处可以只写源文件名，其中源文件的扩展名是需要的。如果目的路径在当前路径下的子路径，那么 dest 可以仅写目的文件夹名称，如欲复制当前路径下的源文件 AA.dat 到当前路径下的子路径 P1 下，那么可以简单地写为 copyfile('AA.dat','\P1') 或者 copyfile('AA.dat','\P1')。当源文件所在路径和目的路径都不是前面所述情况时，source 处需要写全路径信息和文件名，dest 也应该是完整路径名，否则会出现如下提示：

“系统找不到指定的路径。已复制 0 个文件。”

参数 writable 表示确认目的路径是可写属性。status 将返回复制操作执行后的状态，其值等于 0 的时候表示复制操作失败，等于 1 的时候表示复制操作成功。当发生错误时 msg 返回错误信息，如果复制成功，msg 将是一个空字符串。在程序批处理的时候，可以设置两个输出参数 status 和 msg 来判断复制的情况，而且可以保证程序执行不发生中断。

6.7.3 删除文件

函数 delete 可以删除文件，其调用格式为：

```
delete filename       % 格式 1
delete('filename')     % 格式 2
```

filename 应该是完整路径名加完整文件名。如果要删除的文件在当前路径下，那么仅写文件名即可，其中在文件名中扩展名是必须的。如果路径或者文件名中存在空格符，那么需要使用格式 2 的形式删除文件。比如删除 abc.txt 文件可以用下面的语句：

```
delete abc.txt
```

6.7.4 保存命令窗中的会话内容

函数 diary 可以把在命令窗中的会话内容保存到当前路径下的一个文件中，其调用格式如下：

```
diary                       % 格式 1
diary filename              % 格式 2
diary off                   % 格式 3
diary on                    % 格式 4
```

格式 1 将创建一个用户键盘输入并由系统应答的日志文件，同时输出一个包含报告和其他文档的能够打印的 ASCII 文件。格式 2 将把会话内容保存到指定的文件中，如果该文件已经存在，则自动把输出结果直接添加到该文件的结尾处。格式 3 将暂停执行函数 diary。格式 4 恢复执行函数 diary，并使用当前文件名。如果没有指定日志文件的名称，系统将使用默认的日志文件名 diary。用户输入下面的内容可以把用户在命令窗输入的内容记录到 diary_text.txt 文件中。

```
diary diary_text.txt
```

6.7.5 指定路径下的所有文件名

函数 dir 将列出当前或者指定路径下所有文件名，该函数调用格式如下：

```
dir                  % 格式 1
dir dirname            % 格式 2
S = dir('dirname')      % 格式 3
Ss = dir('.ext')         % 格式 4
```

格式 1 将列出当前目录中的所有文件名称及文件夹名称，其中包括隐藏的文件。格式 2 可以用来获取非当前路径下的所有文件信息。格式 3 可以用于文件名或者路径名中包含空格符的情况，S 是一个结构数组，其内部字段含义是：name 为文件名，date 是修改数据，bytes 是文件占用空间的字节数，isdir 等于 1 时表示 name 是个文件夹，否则是一个文件。格式 4 可以获得所有扩展名为某一特定扩展名的文件信息，进一步地我们用 Sn = char(Ss.name)就可以得到文件名信息，其中 Sn 的各行表示文件名信息，用 Sn (k,:)就可以逐一访问这些特定的文件。用户在命令窗中输入 A = dir 将会显示如下内容：

```
A =
75x1 struct array with fields:
   name
   date
   bytes
   isdir
```

```
datenum
```

6.7.6 编辑一个文件

函数 edit 可以编辑一个文件，该函数的调用格式如下：

```
edit                  % 格式 1
edit fun              % 格式 2
edit file.ext         % 格式 3
```

格式 1 将打开一个新的编辑器窗口，作用相当于在 meditor 窗口中单击 New M-file 按钮增加一个新文件。格式 2 将使用编辑器打开指定的 fun.m 文件或者私有文件。格式 3 将打开指定的文本文件。此外，使用 Windows 操作系统的用户可以从 File 菜单中选择 New M-file 命令或者利用函数 Open 来启动 MATLAB 编辑器。在工具栏中单击 New M-file 按钮或者 Open file 按钮也可以打开 MATLAB 编辑器。比如用户在命令窗中输入：

```
edit ode45
```

将会在 M 文件编译窗口弹出 ode45.m 文件。

6.7.7 文件各个部分的信息

函数 fileparts 可以返回文件名各个部分的信息，其调用格式为：

```
[pathstr, name, ext, ver] = fileparts(file)
```

参数说明： pathstr 是文件所在的路径。name 是文件名。ext 是文件扩展名。ver 是版本信息。

比如执行如下语句：

```
[pathstr, name, ext, ver] = fileparts('mesage.kml')
```

将会得到如下输出：

```
pathstr =     ''
name =mesage
ext =.kml
ver =     ''
```

6.7.8 建立完整的文件名

函数 fullfile 可以建立一个完整文件名，其调用格式为：

```
F = fullfile(D1, D2, ..., file);
```

参数说明： F 是返回的完整文件名的字符串。D1，D2 等是按顺序排列的路径名，它们要求是字符串格式。file 是含扩展名的文件名。

比如执行如下语句：

```
F = fullfile(matlabroot,'toolbox','matlab','general','Contents.m')
```

将返回一个完整的文件名。

6.7.9 列出内存中的函数名

函数 inmem 将列出内存中的函数名，其调用格式如下：

```
M = inmem;
[M, MEX] = inmem;
[M, MEX, J] = inmem;
```

M 是 P 码缓冲器里的所有函数名。MEX 是当前装载 MEX 文件名列表。J 是当前装载的 Java 类。另外使用语句 clear all 可以清除内存中的函数、MEX 文件及 Java 类。比如在命令窗中输入 inmem 后会显示当前内存中的函数，这里不再列举，读者可以自己查看。

6.7.10 建立新的文件夹

函数 mkdir 可以建立一个新的文件夹，其调用格式如下：

```
[success, message, messageid] = mkdir(parentdir, newdir);
```

参数说明：输出参数 success 等于 1 时表明文件夹 newdir 成功建立，若 success 等于 0 表示发生错误。message 记录 mkdir 执行后的信息，如果成功建立了文件夹，那么 message 是一个空字符串，否则将返回出错信息。messageid 将给出出错或者警告信息，如果文件夹成功建立，那么 messageid 是空字符串。在使用的时候 3 个输出参数可以全部缺省。如果要建立的文件夹存在，那么 success 将等于 1 同时给出一个警告，通知使用者该文件夹已经存在。parentdir 是所建文件夹的上一级路径信息。newdir 是要建立的文件夹名称。如果 parentdir 不存在，那么 mkdir 函数将按 parentdir 描述顺序依次建立文件夹，并生成新文件夹 newdir。特别地，语句 mkdir('D\AA', 'BB')的结果将是在当前路径下建立 D，再生成子文件夹 AA 及下一级文件夹 BB。因此，使用者在输入 parentdir 信息时一定要准确，否则会引起不必要的麻烦。比如，mkdir('ABC')将在当前路径生成文件夹 ABC。

6.7.11 记录当前路径信息

函数 pwd 可以记录当前路径信息，其调用格式为：

```
S = pwd;
```

参数说明：S 是当前路径的一个字符串，这个函数在切换路径时非常有用。

6.7.12 删除一个路径

函数 rmdir 可以删除一个路径。rmdir 函数操作成功的前提是该路径下应该没有文件，其调用格式为：

```
[success, message, messageid] = rmdir(directory);
```

参数说明：3 个输出参数的意义与 mkdir 输出参数意义类似。directory 是要删除的路径。当要删除的文件夹非空时，将会出现如下的提示：

```
??? Error using ==> rmdir
目录不是空的。
```

使用者可以根据 message 里面的信息考虑清空该文件夹下的文件。

6.7.13 显示 M 文件的全部内容

函数 type 可以把 M 文件的全部内容显示于命令窗内，其调用格式为：

```
type mfilename
```

参数说明：对于当前路径和 MATLAB 搜索路径内的文件，使用 type 函数都可以在命令窗中查看文件的内容。

6.7.14 列出当前路径下的内容

函数 what 可以列出当前路径下的内容，如在命令窗中输入 what 将会出现如下用下画线标示的信息。

```
>> what
M-files in the current directory E:\Program\book1\ch06
script_file
```

6.7.15 基本结构

在进行数据计算的时候，有些大型程序在计算一组数据时，需要很长时间，在程序计算过程中可以去做其他事情。当有多组数据需要保存的时候，可以使用程序自动的保存计算结果功能，因此对于相似文件的批量读写，用户将是非常方便的。

在循环结构中，变化的指标 k（如 for k = 1:K; … end）是 double 型数据，而文件名是字符串型数据。为了统一数据类型，可以使用 num2str，int2str 和 mat2str 来把 double 型数据转为字符串型数据格式。一个通用的批量读、写程序模板如下：

```
for k = 1:N;
    Rd = load(['Data',num2str(k),'.mat']);        % 读入数据
    Rp = imread(['Pic',num2str(k),'.png']);       % 读入图片
    Rm = aviread(['Movie',num2str(k),'.avi']);   % 读入.avi 视频文件
    Rs = wavread(['Music',num2str(k),'.wav']);   % 读入.wav 音频文件
end
for k = 1:N; % 批量写入名称有规律的文件模板如下
    Wd = save(['Data',num2str(k),'.mat'], X);       % 写入数据
    Wp = imwrite(X, ['Pic',num2str(k),'.tif']);        % 写入图片
    Wm = avifile(['Movie',num2str(k),'.avi']);      % 写入.avi 视频文件
    Ws = auwrite(X, ['Music',num2str(k),'.au']);    % 写入.au 音频文件
end
```



对于不同类型的文件，用户可以对读写文件的函数做相应改动。for 循环也可以换为 while 循环。

当使用某一软件不停地产生某一类型的数据，同时对这些数据需要实时处理，而这个软件又无法完成数据处理工作时，如果利用 MATLAB 针对相应文件夹下的数据进行实时处理将是非常便利的，两个软件可以同时运行。下面介绍实时处理数据的一种程序结构：

```
path='D:\Liu\';                    % 指定路径
while 1
    lc = dir([path,'*.dat']);    % 利用通配符获取目标文件名
    D = char(lc.name);        % 分类出文件名
    if ~isempty(D);           % 检测D是否为有效文件名
       y=load(D(1,:));        % 读入数据
       %数据处理部分(用户可以在这里输入自己需要的计算程序)
       delete(D(1,:));        % 删除处理过的文件
    end
    pause(0.1);             % 停顿一下
end
```



在这个结构中，每处理一个文件后必须要把处理过的文件删除，避免出现一直处理该文件的情况。此外也可以利用前面介绍的 copyfile 函数把文件复制到另外一个文件夹下，这样就可以备份数据文件，这个语句应该写在 delete 语句之前。关于数据读入的函数，可以根据实际的数据来选择，有的特殊文件可能需要用户自己去编写一个读数据的小程序。这个程序执行之后可以不停地处理指定文件夹下的数据文件，直至用户手动停止程序。

6.7.16 无规则文件名的处理

当文件名没有规律的时候，可以利用 dir 函数获取目录下任务需求的文件名信息。如果文件名的扩展名确定，可以用下面的语句得到所有文件名：

```
AL = dir(['*.', 'ext']);
Name = char(AL.name);
```

其中 ext 表示扩展名。Name 就是需要的文件名信息。除了利用循环指标命名之外，还可以考虑利用随机数来命名。比如数据写入程序如下：

```
for k = 1:K
   % 计算程序
   Wd = save(['Data', int2str(rand*1e4), '.mat']); % 写入数据
End
```

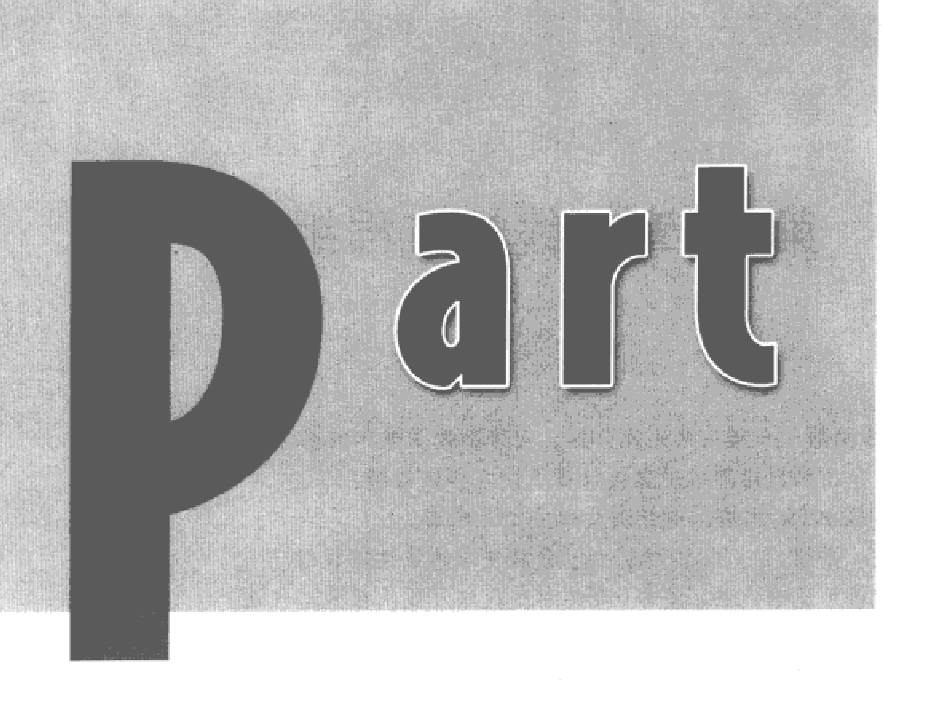
这里利用 int2str(rand*1e4)代替上一节中的 num2str(k)来获取一个随机数。对于其他类型文件的保存也可以类似地做到。

6.8 小结

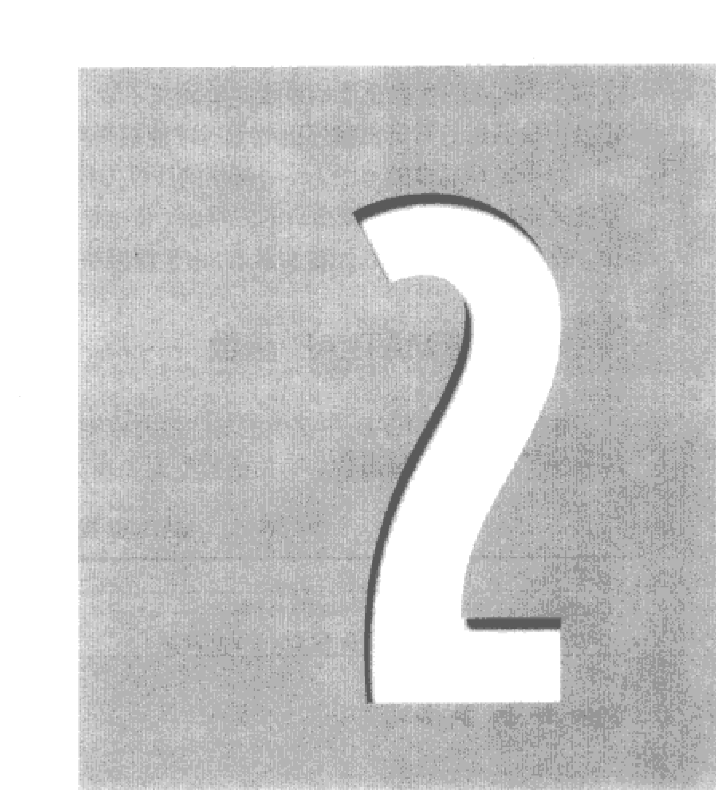
本章主要介绍了一些类型文件的管理与操作知识。首先介绍了脚本文件的定义及一些特点。函数文件是 MATLAB 一个比较有特色的部分，具体介绍了函数文件的定义、inline 函数的使用方法、分段函数的定义方法、子函数的使用、私有函数的定义和管理等。函数文件和脚本文件各有各自的特点，需要根据它们的特点选择使用。数据文件的处理是经常用到的，这里详细介绍了数据文件读

写操作的相关知识。接下来介绍了图片文件、视频文件和音频文件的读写操作。为了方便用户处理大量文件，本章最后还详细介绍了文件批量处理的程序结构。

◆ ◆ ◆



第2部分 科学计算



第 7 章　线性方程组

本章包括

- ◆ **基础 MATLAB 函数**　介绍一些矩阵分解、矩阵参数及操作的基本函数。
- ◆ **矩阵求逆法**　介绍利用矩阵求逆直接计算线性方程组的解。
- ◆ **消元法**　介绍 Gauss 消元法、追赶法及列主元消元法。
- ◆ **矩阵分解算法**　介绍 LU，QR 和 cholesky 分解法求解线性方程组。
- ◆ **迭代法**　介绍雅可比迭代法、高斯-赛德尔迭代法及逐次松弛迭代法。
- ◆ **共轭梯度法**　介绍相关 MATLAB 函数的使用方法。

求线性方程组的问题，不但在科学技术中有所涉及，而且在计算方法其他分支的研究中，比如样条插值、最佳平方逼近、微分方程数值解等，也往往需要解线性方程组，因此这是一个应用相当广泛的分支。本章介绍用 MATLAB 解线性方程组的方法。MATLAB 软件最早起源于对线性方程组的研究。

线性方程组的数值解法通常分为两类：直接法和迭代法。直接法是在没有舍入误差的情况下，通过有限步四则运算可以求得方程组。但是在实际计算时，由于初始数据变为机器数而产生的误差以及计算过程中所产生的舍入误差等都对解的精确度产生影响，因此直接法实际上也只能算出方程组的近似解。目前较实用的直接法是 Gauss 消元法和一些变形，例如选主元的 Gauss 消元法和矩阵的三角分解法，它们都是目前计算机上常用的有效方法。直接法的优点是计算量小，并且可事先估计，缺点是要求存储空间较多，编写程序较为复杂。

迭代法是用某种极限过程去逐步逼近准确解的方法，从而也可用有限步运算得出具有指定精确度的近似解。迭代法主要有：Jacobi 迭代法、Gauss-Seidel 迭代法、逐次超松弛法以及共轭斜量法等。迭代法的优点是原始系数矩阵始终不变，因而算法简单，编写程序方便，且所需存储空间也较少，缺点是只有近似解序列收敛时才能被采用，而且存在收敛性和收敛速度的问题。

对于中等规模的 n 阶（$n<100$）的线性方程组，由于直接法的准确性和可靠性，所以它们是经常被采用的方法。对于较高阶的方程组，特别是低于某些偏微分方程离散化后得到的大型稀疏方程组（系统矩阵绝大多数为零元素），由于直接解法的计算代价较高，使得迭代法更具有竞争力。

7.1　基础 MATLAB 函数

本节介绍 MATLAB 中提供的与线性方程组相关的函数及使用方法，其中包括矩阵的创建、矩阵分解、矩阵操作等函数。有关函数整理如表 7.1 所示。

表 7.1　MATLAB 提供的与矩阵有关的函数

函数名	说明	函数名	说明
chol	矩阵的 cholesky 分解	lu	矩阵的 LU 分解
cholinc	矩阵的不完全 cholesky 分解	luinc	矩阵的不完全 LU 分解

（续表）

函数名	说明	函数名	说明
diag	提取矩阵对角元素	norm	矩阵或矢量的范数
eig	求本征值和本征向量	normest	求矩阵的最大范数矢量
eye	生成单位矩阵	pinv	求伪逆矩阵
fliplr	左右翻转矩阵	qr	QR 分解
flipud	上下翻转矩阵	rank	求矩阵的秩
inv	求逆矩阵	trace	矩阵对角元素的和（迹）

下面逐一介绍这些函数的使用。

7.1.1 矩阵的 cholesky 分解

函数 chol 进行矩阵的 cholesky 分解，即求解上三角矩阵 R 使 R'*R=X。其调用格式为：

```
R = chol(X);       % 格式 1
[R, p] = chol(X);   % 格式 2
```

使用说明：如果矩阵 X 正定，可以使用格式 1。如果矩阵 X 非正定，那么需要使用格式 2（直接使用格式 1 会提示出错），参数 p 表示子矩阵 X(1:p-1,1:p-1)是正定的，即 R'*R= X(1:p-1,1:p-1)。

7.1.2 矩阵的不完全 cholesky 分解

函数 cholinc 实现矩阵的不完全 cholesky 分解，即求出上三角矩阵 R 使 X = R'*R-A。其调用格式为：

```
R = cholinc(X,droptol);
R = cholinc(X,options);
R = cholinc(X,'0');
[R,p] = cholinc(X,'0');
R = cholinc(X,'inf');
```

参考说明：矩阵 X 要求是稀疏矩阵的形式，可以使用函数 sparse 把 double 型矩阵转为稀疏矩阵形式。droptol 用来指定误差。options 取值丰富多彩，其中，droptol 表示舍入误差；如果 michol=1，则从对角线上抽取出已去掉的元素；rdiag 表示用 droptol 值代替上三角分解因子中对角线上的零元素。0 是一种分解标准。如果输出含有参数 p，那么不产生任何出错信息，如果 R 存在，则 p=0；如果 R 不存在，则 p 为正整数。inf 表示进行 cholesky 无穷分解。

7.1.3 提取矩阵的对角元素

函数 diag 可以用来提取矩阵的对角元素，同时它还可以实现其他相关功能。其调用格式如下：

```
X = diag(v,k);          % 格式 1
X = diag(v);            % 格式 2
v = diag(X,k);          % 格式 3
v = diag(X);            % 格式 4
```

使用说明：X 和 v 分别表示矩阵和向量。格式 1 中由向量 v 生成 n×n 方矩阵 X，n=length(v)+abs(k)，其中 k 表示离开主对角线的斜线数（k 为正整数时在主对角线上面，k 为负

整数时在主对角线下面)，矩阵 X 中离开主对角线的斜线由向量 v 赋值。k 的默认值是 0。k 可以取小数，但是 MATLAB 按向中心取整处理。格式 3 是提取矩阵离开主对角线的 k 斜线上的向量。

7.1.4 求本征值和本征向量

函数 eig 可以用来求本征值和本征向量。一般本征值用来求解方程 $Ax = \lambda x$，广义本征值用来求方程 $Ax = \lambda Bx$ 的解。函数 eig 的调用格式如下：

```
d = eig(A);
d = eig(A,B);
[V,D] = eig(A);
[V,D] = eig(A,'nobalance');
[V,D] = eig(A,B);
[V,D] = eig(A,B,flag);
```

参考说明： d 是列向量，V 是本征向量矩阵（其中列向量对应矩阵 A 的本征向量），D 是由本征值构成的对角矩阵。nobalance 表示直接求矩阵 A 的特征值和特征向量，没有这个参数的时候先对 A 进行相似变换后求矩阵 A 的特征值和特征向量。当表达式中含参数 B 的时候，函数 eig 计算广义本征值向量矩阵 V 和广义本征值矩阵 D，满足 AV=BVD。参数 flag 用来指定算法计算特征值 D 和特征向量 V，flag 的值为 chol 表示对 B 使用 cholesky 分解算法，这里 A 为对称 Hermitian 矩阵，B 为正定阵；flag 的值为 qz 表示使用 QZ 算法，这里 A 和 B 为非对称或非 Hermitian 矩阵。

7.1.5 矩阵的基本运算

在 MATLAB 中，非常方便地进行矩阵的基本运算，包括生成单位矩阵、矩阵的翻转、矩阵的求逆等。下面分别对这些基本运算进行讲解。

函数 eye 可以生成单位矩阵，其调用格式如下：

```
Y = eye(n);
Y = eye(m,n); 或者 Y = eye(size(A));
```

弹出的警告信息如下：

```
Warning: Size vector should be a row vector with integer elements.
```

函数 fliplr 可以实现左右翻转矩阵，其调用格式为：

```
B = fliplr(A);
```

参数说明： 当 A 是列向量时 B=A。

函数 flipud 可以实现上下翻转矩阵，其调用格式为：

```
B = flipud(A);
```

参数说明： 当 A 是行向量时 B=A。

函数 inv 可以用来求逆矩阵，其调用格式为：

```
Y = inv(X)
```

参数说明： X 必须是方矩阵，同时函数 inv 支持符号矩阵求逆。

7.1.6 矩阵的 LU 分解

函数 lu 可以用来求矩阵的 LU 分解，其调用格式为：

```
[L, U] = lu(X);                      %对应于 L*U=X
[L, U, P] = lu(X);                   %对应于 L*U=P*X
Y = lu(X);
[L, U, P, Q] = lu(X);                %对应于 L*U=P*X*Q
[L, U, P] = lu(X, thresh);           %对应于 L*U=P*X
[L, U, P, Q] = lu(X, thresh);        %对应于 L*U=P*X*Q
```

参数说明： X 是任意矩阵。矩阵 Y 可以表示为 Y = U+L-speye(size(X))，其中[L, U, P] = lu(X)。U 为上三角阵，L 为下三角阵或其变换形式。P 为单位矩阵的行变换矩阵，Q 为单位矩阵的列变换矩阵。thresh 是中心阈值，其值分布于区间[0,1]，默认值为 1。

7.1.7 矩阵的不完全 LU 分解

函数 luinc 可以实现矩阵的不完全 LU 分解，其调用格式为：

```
[L, U] = luinc(X, '0');
[L, U, P] = luinc(X, '0');
[L, U] = luinc(X, options);
[L, U] = luinc(X, droptol);
[L, U, P] = luinc(X, options);
[L, U, P] = luinc(X, droptol);
```

参数说明： 矩阵 X 要求是一个稀疏方矩阵。U 为上三角阵，L 为下三角阵或其变换形式。P 为单位矩阵的行变换矩阵。0 是一种分解标准。droptol 指定不完全分解的舍入误差。options 取值为：droptol 表示指定的舍入误差；milu 表示改变分解以便于从上三角分解因子中抽取出已去掉的列元素；ugiag 为 1 表示用 droptol 值代替上三角因子中对角线上的零元素，其默认值为 0；thresh 为中心临界值。

7.1.8 矩阵范数的计算

函数 norm 可以计算矩阵或矢量的范数，其调用格式为：

```
n = norm(A);
n = norm(A,p);
```

参数说明： n 表示返回的范数。A 是待求矩阵或者向量。p 用来控制计算不同的范数：p=1 时计算 1-范数，返回矩阵 A 各列绝对值之和的最大值，即等价于 max(sum(abs(A))；p=2 时计算 2-范数，也称为最大奇异值；p=inf 时计算 ∞-范数，等于 A 的行向量的 1-范数的最大值，即 max(sum(abs(A'))); p='fro'时求矩阵 A 的 Frobenius 范数，即 sqrt(sum(diag(A'*A)))。

函数 normest 可以用来求矩阵的最大范数矢量，其调用格式为：

```
nrm = normest(S);
nrm = normest(S, tol);
[nrm, count] = normest(S, tol);
```

参数说明： nrm 表示返回的范数矢量。S 是输入矩阵。tol 为指定的相对误差，其默认值为 1e-6。count 是计算范数迭代的次数。

7.1.9 计算伪逆矩阵

函数 pinv 可以用来计算伪逆矩阵，当一个矩阵是非方矩阵的时候，只能计算其伪逆矩阵。其调用格式为：

```
B = pinv(A);
B = pinv(A,tol);
```



B 为返回的伪逆矩阵。tol 是误差，默认值为 max(size(A))*norm(A)*eps。当 A 为方矩阵的时候，inv(A)和 pinv(A)返回的结果相同。

7.1.10 矩阵的 QR 分解

函数 qr 实现矩阵的 QR 分解，即将矩阵分解成一个正交矩阵与一个上三角矩阵的乘积。该函数的调用格式为：

```
[Q,R] = qr(A);              %A 为 double 型矩阵或者稀疏矩阵
[Q,R] = qr(A,0);            %A 为 double 型矩阵或者稀疏矩阵
[Q,R,E] = qr(A);            %A 为 double 型矩阵，不能是稀疏矩阵
[Q,R,E] = qr(A,0);          %A 为 double 型矩阵，不能是稀疏矩阵
X = qr(A);                  %A 为 double 型矩阵，不能是稀疏矩阵
R = qr(A);                  %A 为稀疏矩阵，R'*R = A'*A
[C,R] = qr(A,B);            %A 为稀疏矩阵，其中 C = Q'*B
R = qr(A,0);                %A 为稀疏矩阵，针对稀疏矩阵 A 的经济型分解
[C,R] = qr(A,B,0);          %A 为稀疏矩阵，针对稀疏最小二乘问题的经济型分解
```

参数说明： A 为输入矩阵，其可以选择 double 型矩阵和稀疏矩阵。Q 是正交矩阵，R 是上三角矩阵。X 是利用 LAPACK 库函数得到的矩阵，其中 triu(X)=R。

7.1.11 计算矩阵的秩与迹

函数 rank 可以用来求矩阵的秩，其调用格式为：

```
k = rank(A);
k = rank(A,tol);
```

参数说明： 参数 A 是输入的待求矩阵，其不能是稀疏矩阵形式。k 是计算得到矩阵的秩。tol 是指定的误差，其默认值是 max(size(A))*norm(A)*eps。

利用函数 trace 可以得到矩阵对角元素的和（迹），其调用格式为：

```
b = trace(A)
```

参数说明： trace 函数等价于 sum(diag(A))和 sum(sum(A.*eye(size(A))))。

7.2 矩阵求逆法

为了理解矩阵求逆的方法，首先介绍线性方程组 AX=B 的基本理论。

- n 阶行列式 A：|A|等于所有取自不同行不同列的 n 个元素的积的代数和。
- 矩阵 $\boldsymbol{B}$：是一个列向量。
- 线性无关：一列向量组$(a_1,a_2,\cdots,a_n)$不与线性相关，即没有不全为零的系数$c_1,c_2,\cdots,c_n$使得$c_1a_1+c_2a_2+\cdots c_na_n=0$成立。
- 秩：向量组的极大线性无关组所含向量的个数称为这个向量组的秩，记为 R(A)。
- 一般线性方程组 AX=B 是指如下形式：

$$\begin{cases} a_{11}*x_1+a_{12}*x_2+\ldots+a_{1n}*x_n=b_1 \\ a_{21}*x_1+a_{22}*x_2+\ldots+a_{2n}*x_n=b_2 \\ \qquad\qquad \cdots \\ a_{m1}*x_1+a_{m2}*x_2+\ldots+a_{mn}*x_n=b_m \end{cases}$$

其中，x_1，x_2，$\cdots$，x_n为 n 个未知数，m 为方程个数。记：A*X=b。

- 方程组 AX=B 的增广矩阵为：

$$\bar{A}：\bar{A}=\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & b_m \end{pmatrix}$$

- A*X=0 是方程组 AX=B 对应的齐次线性方程组。
  对于线性方程组有 3 种基本变换，即：
  - 用非零数两边同乘其中某一个方程。
  - 将一个方程倍数后加到另一个方程。
  - 互换两个方程的位置。

  以上的操作称初等变换。利用这 3 种初等变换可以把方程转化为比较简单的等价形式。
  线性方程组有解的必要条件为$R(A)=R(\bar{A})$。

接下来介绍不同类型线性方程组解的结构。

- 齐次线性方程组
  - 两个解之和还是方程组的解。
  - 一个解的倍数还是方程组的解。

  定义：齐次线性方程组的一组解s_1，s_2，$\cdots$，s_k称为齐次线性方程组的一个基础解系，如果齐次线性方程组的任一解都能表示成s_1，s_2，$\cdots$，s_k的线性组合，则s_1，s_2，$\ldots$，s_k线性无关。
- 非齐次线性方程组

- 方程组 AX=B 的两个解的差是 AX=0 的解。
- 方程组 AX=B 的一个解与 AX=0 的一个解之和还是 AX=B 的解。

定理：如果 x_0 是方程组 AX=B 的一个特解，那么方程组 AX=B 的任意一个解 x 都可以表示成 $x=x_0+cv$。其中 v 是 AX=0 的一个解，c 是系数，因此对方程 AX=B 的任一特解为 x_0，当 v 取遍方程组 AX=0 的全部线性无关解后，$x=x_0+cv$ 就给出了方程组 AX=B 的全部解。

线性方程的求解分为 3 类：一类是方程组求唯一解；一类是求特解；一类是在特解基础上求无穷解，即通解。具体求解步骤如下。

step 1 判断方程组解的情况。

- 当 $R(A)=R(\bar{A})$ 时，若 $R(A)=R(\bar{A})=n$，则有唯一解，若 $R(A)=R(\bar{A})<n$ 则有无穷多解。
- 当 $R(A)<R(\bar{A})$ 时无解。

step 2 求特解。特解的计算可以用 X=A\B（或者用函数 pinv 求伪逆矩阵来计算特解 X=pinv(A)*B）计算。

step 3 求通解（无穷多解），AX=B 的无穷多解为 AX=0 的通解加 AX=B 的一个特解。



以上针对非齐次线性方程组 AX=B。齐次线性方程组 AX=0，则主要是用到步骤 1 和步骤 2。

对于线性方程组 AX=B 或 XA=B，在 MATLAB 中求解时，可以直接采用矩阵的除法运算符"/"和"\"来实现。如：

- X=A\B 表示求线性方程组 AX＝B 的解。
- X＝B/A 表示求线性方程组 XA=B 的解。

这里，对于方程组 X＝A\B，要求 A 和 B 有相同的行数，X 和 B 有相同的列数，它的行数等于矩阵 A 的列数，方程 X＝B/A 同理。事实上，可以通过计算 inv(A)*B 和 B*inv(A)来分别计算线性方程组 AX＝B 和 XA=B 的解。

例 7-1： 求下面的线性方程组的解。

$$\begin{cases} x_1+2x_2+3x_3=1 \\ 6x_1+x_2+2x_3=2 \\ 4x_1+3x_2+2x_3=3 \end{cases}$$

求解程序如下：

```
A = [1,2,3;6,1,2;4,3,2];          % 输入方程系数矩阵 A
B = [1;2;3];                      % 向量 B
X = A\B                           % 直接求解方程的解
X = X'                            % 求转置矩阵
```

输出结果如下：

```
X = 0.3000    0.8000   -0.3000
```



可以通过 AX−B 来验证解的正确性。

要指出的一点是，在使用“X=A\B”求方程组的解之前，最好利用 det 或者 rank 函数查看一下方程是否有唯一解。

例 7-2：求解下面的方程组，其中 A=magic(4)，B=[1,1,1,1]'。

$$AX = B$$

在 MATLAB 中可以用下面的语句先得到方程组的解：

```
A = magic(4);              % 输入系数矩阵
B = [1, 1, 1, 1]';         % 输入向量 B
X0 = [A\B]'                % 求解方程
```

输出结果如下：

```
Warning: Matrix is close to singular or badly scaled.
        Results may be inaccurate. RCOND = 1.306145e-017.
(Type "warning off MATLAB:nearlySingularMatrix" to suppress this warning.)
> In E:\Program\book1\ch07\example_7_1.m at line 11
X0 =0.0588    0.1176   -0.0588         0
```

这说明系数矩阵 $\boldsymbol{A}$ 是奇异的，方程组解的情况需要进一步计算才能确定，下面计算矩阵 $\boldsymbol{A}$ 和 [A, B]的秩。

```
r1 = rank(A);
r2 = rank([A, B])
```

如果 r1=r2 说明方程有无穷多解，否则方程无解。在这个例子中，r1 = r2 = 3<4，说明方程有无穷多解，而前面得到的 X0 只是一个特解，通过[A*X0'-B]'验证有：

```
ans =  1.0e-015 *
        0          0          0   -0.2220
```

表明 X0 是方程组的解，它可以作为一个特解。通解的计算步骤如下：

step 1 求 AX=0 的通解。

step 2 AX=b 的通解为 AX=0 的通解加上 AX=b 的一个特解。

方程组 AX=0 的通解可以使用 MATLAB 提供的函数 null 来计算，其调用格式为：

```
X = null(A);
X = null(A, 'r');
```

参数说明：X 是返回的通解。A 是输入的矩阵。参数 r 表示如果 A 的系数是整数，X 将是由整数组成的矩阵。

使用下面的命令可以得到方程组 AX=0 的通解。

```
Xn1 = null(A)   % 求出解空间的一组基（基础解系）
```

```
Xn2 = null(A,'r') % 求出解空间的一组基（基础解系）
```

根据上面命令得出的结果是：

```
Xn1 =
    0.2236
    0.6708
   -0.6708
   -0.2236
Xn2 =
    -1
    -3
     3
     1
```

可以发现利用参数 r 得到整数通解。通解的构造公式为：$X_c = X_0 + C_1x_1 + ... + C_rx_r$，其中 X_0 是特解，C_1，…，C_r 是任意实数，x_1，…，x_r 是利用 null 函数得到矩阵（r 是矩阵的列数）的列向量。

如果把例 7-2 中的矩阵 B 换为 B=[1: 4]'，计算 X=A\B 有：

```
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 1.306145e-017.
(Type "warning off MATLAB:nearlySingularMatrix" to suppress this warning.)
> In E:\Program\book1\ch07\example_7_1.m at line 10
X =  1.0e+015 *
   -0.5629   -1.6888    1.6888    0.5629
```

可以检查所得解 X 是否满足方程，即计算 A*X'-B，输出如下：

```
ans =   -1.0000
         4.5000
         1.0000
         3.8125
```

可见得到的解其实不是方程组的解，也不必再求解下去，即可以判定方程组无解。

例 7–3：利用函数 rref 求下面方程组的解。

$$\begin{cases} x_1 - x_2 + x_3 - x_4 = 1 \\ -x_1 + x_2 + x_3 - x_4 = 1 \\ 2x_1 - 2x_2 - x_3 + x_4 = -1 \end{cases}$$

分析：直接观察方程可知，方程个数少于未知数个数，因此方程可能是无解或者有无穷多解。

```
A = [1,-1,1,-1;-1,1,1,-1;2,-2,-1,1];          % 输入系数矩阵 A
B = [1;1;-1];                              % 输入向量 B
rr = [rank(A), rank([A,B])]                  % 求系数矩阵 A 和增广矩阵的秩
```

输出如下：

```
rr =2
```

因为系数矩阵和增广矩阵的秩相等且小于未知数的个数，所以方程有无穷多解。进一步调用函数 rref。

```
C = rref([A,B])                          % 求增广矩阵行的最简阶梯形矩阵形式
```

得到的输出结果为：

```
C =     1    -1     0     0     0
        0     0     1    -1     1
        0     0     0     0     0
```

上式对应于$\begin{cases} x_1 - x_2 = 0 \\ x_3 - x_4 = 1 \end{cases}$，因此通解可以写为：$x_1 = x_2$，$x_3 = x_4 + 1$，其中$x_2$和$x_4$可以在定义范围内任意取值。

7.3 消元法

消元法首先用初等变换转化线性方程组为阶梯形方程组，把最后的一些恒等式“0=0”（如果出现的话）去掉。如果剩下的方程当中最后的一个等式是零等于一个非零数，那么方程组无解；否则有解。在有解的情况下，可以分为如下两个情况：

- 如果阶梯形方程组中方程的个数r等于未知量的个数，那么方程组有唯一的解。
- 如果阶梯形方程组中方程的个数r小于未知量的个数，那么方程组就有无穷多个解。

用初等变换转化线性方程组为阶梯形方程组，相当于用初等行变换转化增广矩阵成阶梯形矩阵。根据阶梯形矩阵就可以判别方程组有解还是无解，在有解的情形下，回到阶梯形方程组去求解。

利用 MATLAB 提供的函数 rref 可以得到求矩阵行的最简阶梯形矩阵形式，该函数调用格式为：

```
R = rref(A);
[R,jb] = rref(A);
[R,jb] = rref(A,tol);
```

参数说明：R 是返回的最简阶梯形矩阵形式。jb 中元素表示基向量所在的列。A 是输入的矩阵，它可以是系数矩阵$\boldsymbol{A}$或增广矩阵$\bar{\boldsymbol{A}}$。tol 为指定的精度，其默认值为(max(size(A))*eps*norm(A,inf))，相应地有函数 rrefmovie 来给出每一步化简的过程。

用下面的语句求矩阵行的最简阶梯形矩阵形式：

```
A=magic(3);         % 系数矩阵
B=[1:3]';           % 向量 B
Ap=[A',B];          % 增广矩阵
rrefmovie(Ap)       % 逐步显示简化结果
C=rref(Ap)          % 求矩阵行的最简阶梯形矩阵形式
```

输出如下：

```
A =
     8     3     4     1
     1     5     9     2
     6     7     2     3
```

使用者按任意键就可以查看每一步的简化结果，其中为了显示方便，rrefmovie 执行的时候含命令窗清屏（即函数 clc）操作，可以在学习线性代数课程时用这个函数做演示示例。

矩阵 Ap 行的最简阶梯形矩阵形式为：

```
C =    1.0000         0         0   -0.0333
       0    1.0000         0    0.4667
       0         0    1.0000   -0.0333
```

由于矩阵 C 前面 3 列对应的矩阵是单位矩阵，可知上面结果最右侧一列就是方程 magic(3)*X=[1;2;3]的解。

首先需要考虑三角方程组的解法。对于上三角形方程组，其矩阵形式表示为 $Ux = B$，这里 U 是上三角矩阵，即：

$$U = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1,n-1} & u_{1n} \\ & u_{22} & \cdots & u_{2,n-1} & u_{2n} \\ & & \ddots & & \vdots \\ & & & u_{n-1,n-1} & u_{n-1,n} \\ & & & & u_{nn} \end{bmatrix} \tag{7-1}$$

x 表示未知数向量。B 表示列向量。若矩阵 U 的行列式不等于 0，即 $u_{ii} \neq 0(i = 1,2,\cdots,n)$，矩阵 U 是非奇异的，则上三角方程组（7-1）有唯一解，且可从最后一个方程解出，即：$x_n = y_n / u_{nn}$。

套入倒数第二个方程得到：

$$x_{n-1} = \left(y_{n-1} - u_{n-1}x_n\right) / u_{n-1,n-1}$$

一般情况，假设已求得 $x_n, x_{n-1}, \cdots, x_{i+1}$，则由方程的第 i 个方程可得：

$$x_i = \left(y_i - \sum_{j=i+1}^{n} u_{ij}x_j\right) \Big/ u_{ii} \qquad i = n-1, n-2, \cdots, 1 \tag{7-2}$$

依次计算下去即可得到方程组的所有未知数。上述求解过程被称为回代过程。回代过程所用乘除法运算次数为 $M_1 = n(n+1)/2$，加减法运算次数为 $S_1 = n(n-1)/2$。根据公式（7-2），编写了函数文件 uelim.m（保存在光盘中）来实现消元法解上三角方程组。

例 7-4：利用消元法求解下面的方程组。

$$\begin{cases} 3x_1 + 2x_2 + x_3 + 4x_4 = 1 \\ \qquad 5x_2 + x_3 + 2x_4 = 9 \\ \qquad\qquad 2x_3 + 5x_4 = 7 \\ \qquad\qquad\qquad 2x_4 = 9 \end{cases}$$

求解的 MATLAB 程序如下：

```
A = [3,2,1,4;0,5,1,2;0,0,2,5;0,0,0,2];        % 输入系数矩阵 A
```

```
B = [1;9;7;9];                    % 输入向量 B
X = uelim(U,B)                    % 用消元法解上三角方程组
```

输出结果如下：

```
X =   -4.1167    1.5500   -7.7500    4.5000
```

可见上三角方程组的求解方法比较简单，但对于一般的线性方程组 AX=B 而言，若能通过变换将其化解为上三角方程组这种特殊形式，则求解方程组的问题也就自然解决了。Gauss 消元法正是实现这一想法的求解方法。

将原线性方程组写成增广矩阵的形式，其中 $a_{ij}^{(1)}=a_{ij}$，$b_{i,n+1}^{(1)}=b_i$，$1\le i,j\le n$。

$$\overline{A}^{(1)}=\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} & b_2^{(1)} \\ \vdots & \vdots & & \vdots & \vdots \\ a_{n-1,1}^{(1)} & a_{n-1,2}^{(1)} & \cdots & a_{n-1,n}^{(1)} & b_{n-1}^{(1)} \\ a_{n1}^{(1)} & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} & b_n^{(1)} \end{bmatrix} \tag{7-3}$$

第 1 步消元：假设 $a_{11}^{(1)}\neq 0$，将 $\overline{A}^{(1)}$ 中的第 1 行乘以 $l_{i1}=-a_{i1}^{(1)}/a_{11}^{(1)}$，加到第 i（$i=2,3,\cdots,n$）行上去，可得到同解方程组的增广矩阵。

$$\overline{A}^{(1)}\xrightarrow[\substack{\vdots \\ r_n+(-l_{n1})r_1}]{\substack{r_2+(-l_{21})r_1 \\ r_3+(-l_{31})r_1}}\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(2)} & \cdots & a_{1n}^{(2)} & b_1^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} & b_2^{(1)} \\ 0 & a_{32}^{(2)} & \cdots & a_{3n}^{(2)} & b_3^{(2)} \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} & b_n^{(2)} \end{bmatrix}=\overline{A}^{(2)}$$

其中 $a_{ij}^{(2)}=a_{ij}^{(1)}-l_{i1}a_{1j}^{(1)}$，$1\le i\le n,2\le j\le n+1$。这里符号 $r_i+(-l_{i1})r_1$ 表示对 $\overline{A}^{(1)}$ 的第 i 行进行变换，将第 1 行的 $(-l_{i1})$ 倍加到第 i 行上。

第 k 步消元：设 $a_{kk}^{(k)}\neq 0$，将 $\overline{A}^{(k)}$ 中的第 k 行对角线以下的元素消为零，即将 $\overline{A}^{(k)}$ 中的第 k 行乘以 $l_{ik}=-a_{ik}^{(k)}/a_{kk}^{(k)}$，加到第 i（$i=k+1,k+2,\cdots,n$）行上去可得到同解方程组的增广矩阵。

$$\overline{A}^{(k)}\xrightarrow[\substack{\vdots \\ r_n+(-l_{nk})r_k}]{\substack{r_{k+1}+(-l_{k+1,k})r_k \\ r_{k+2}+(-l_{k+2,k})r_k}}\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1k}^{(1)} & a_{1,k+1}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2k1}^{(2)} & a_{2,k+1}^{(2)} & \cdots & a_{2n}^{(2)} & b_2^{(2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & a_{kk}^{(k)} & a_{k,k+1}^{(k)} & \cdots & a_{kn}^{(k)} & b_k^{(k)} \\ 0 & 0 & \cdots & 0 & a_{k+1,k+1}^{(k+1)} & \cdots & a_{k+1,n}^{(k+1)} & b_{k+1}^{(k+1)} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & a_{n,k+1}^{(k+1)} & \cdots & a_{nn}^{(k+1)} & b_n^{(k+1)} \end{bmatrix}=\overline{A}^{(k+1)}$$

其中$a_{ij}^{(k+1)}=a_{ij}^{(k)}-l_{ik}a_{kj}^{(k)}$，$k+1\le i\le n,k+1\le j\le n+1$。称参数$l_{ik}$为消元因子，$a_{kk}^{(k)}$为$k$步矩阵$\overline{A}^{(k)}$主元。

上述做法直至第 n-1 步完成，得到同解的方程组，如下：

$$\overline{A}^{(n)}=\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1,n-1}^{(1)} & a_{1,n}^{(1)} & b_1^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2,n-1}^{(2)} & a_{2,n}^{(2)} & b_2^{(2)} \\ 0 & \cdots & \ddots & \vdots & \vdots & \vdots \\ \vdots & 0 & \cdots & a_{n-1,n}^{(n-1)} & a_{n=1,n}^{(n-1)} & b_{n-1}^{(n-1)} \\ 0 & \cdots & \cdots & 0 & a_{nn}^{(n)} & b_n^{(n)} \end{bmatrix} \tag{7-4}$$

$\overline{A}^{(n)}$为上三角矩阵，这时用回代过程便可求得解向量x。



如果出现$a_{kk}^{(k)}\ne 0$的情况，可以把第k行与下面的某一行互换，以保证$a_{kk}^{(k)}\ne 0$。

上述的 Gauss 消元过程中，乘除法次数为：$M_2=\frac{1}{3}n^3+\frac{1}{2}n^2-\frac{5}{6}n$，$M=M_1+M_2=\frac{1}{3}n^3+n^2-\frac{1}{3}n$；加减法次数为：$S_2=\frac{1}{3}n^3-\frac{1}{3}n$，$S=S_1+S_2=\frac{1}{3}n^3+\frac{1}{2}n^2-\frac{5}{6}n$。

Gauss 消元法可分为消元和回代两个过程，于是 Gauss 消元法总的运算量为：乘除法次数$M=M_1+M_2$加加减法次数$S=S_1+S_2$。由于在计算机运算中，做一次乘除所花费的时间大大超过做一次加减法所需的时间，因此估计某个方法所需运算量时，往往只需估计乘除次数，即 Gauss 消元法的运算量为M。当n很大时，可以略去n的低次幂项（n^2和n），这样 Gauss 消元法的计算量为$n^3/3$数量级。例如当$n=30$时，Gauss 消元法需做 9890 次乘除法。

利用 MATLAB 提供的函数 qr 可以把矩阵 A 分解为正交矩阵 Q 和上三角矩阵 U。利用正交矩阵的性质，可以用矩阵 Q 的转置矩阵 Q'同乘方程 AX=B 的两侧而得到一个上三角方程。这样就免去了按上述过程编写 Ganss 消元程序。

在实际应用中存在着一类三对角形方程组，如样条函数的计算、微分方程数值求解等。这种方程可以写为如下形式：

$$\begin{bmatrix} b_1 & c_1 & & & \\ a_2 & b_2 & c_2 & & \\ & \ddots & \ddots & & \\ & & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & a_n & b_n \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix}=\begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix} \tag{7-5}$$

系数矩阵中非零元素分布在主对角线及相邻的两条斜线上，同时需元素满足：

- $|b_1|>|c_1|>0$
- $|b_i|\ge|a_i|+|c_i|,\ a_ic_i\ne 0\ (i=2,3,\cdots,n-1)$

◆ $|b_n|>|a_n|>0$

利用方程组（7-5）的特点，应用 Gauss 消元法求解时，在每一步消元中只需消去一个元素。消元过程为：

$$\begin{cases}\beta_1 = b_1, & y_1 = d_1 \\ l_i = \dfrac{a_i}{\beta_{i-1}}, & y_i = d_i - l_i y_{i-1},\ i = 2,3,\cdots,n\end{cases} \tag{7-6}$$

得到同解方程组为：

$$\begin{bmatrix}\beta_1 & c_1 & & & & y_1 \\ & \beta_2 & c_2 & & & y_2 \\ & & \ddots & \ddots & & \vdots \\ & & & \beta_{n-1} & c_{n-1} & y_{n-1} \\ & & & & \beta_n & y_n\end{bmatrix} \tag{7-7}$$

回代过程为：

$$\begin{cases}x_n = y_n / \beta_n, \\ x_i = (y_i - c_i x_{i+1}) / \beta_i,\ i = n-1, n-2, \cdots, 1\end{cases} \tag{7-8}$$

这种把三对角方程组（7-5）的解用递推公式（7-7）和（7-8）表示出来的方法被形象地叫做追赶法。公式（7-6）是下标 i 由小到大的递推公式，故被称为追的过程。公式（7-8）是关于下标 i 从大到小的递推过程，故被称为赶的过程。用追赶法解方程组（7-5）仅需要$(5n-4)$次乘除过程和$(3n-3)$次加减过程。

在前面介绍的 Gauss 消元法中，只有在 $a_{kk}^{(k)} \neq 0\ (k=1,2,\cdots,n-1)$ 下才能进行，而且需指出即使 $a_{kk}^{(k)} \neq 0$，当 $\left|a_{kk}^{(k)}\right|$ 和 $\left|a_{ik}^{(k)}\right|\ (k+1 \le i \le n)$ 相比很小时，也是不适用的。因为在第 k 步消元时，需将 $\overline{A}^{(k)}$ 的第 k 行乘以 $(-l_{ik})$ 加到第 i 行。如果第 k 行的元素 $\left(a_{k,k+1}^{(k)}, a_{k,k+2}^{(k)}, \cdots, a_{k,k+n}^{(k)}\right)$ 有误差 $(\varepsilon_{k+1}, \varepsilon_{k+2}, \cdots, \varepsilon_{n+1})$，则该误差将放大 $(-l_{ik})$ 倍传到 $\overline{A}^{(k)}$ 的第 i 行。这样 $l_{ik}\left(=a_{ik}^{(k)}/a_{kk}^{(k)}\right)$ 的绝对值很大，由此将带来舍入误差的严重增长。

解决这一问题的方法之一是选列主元。假设已完成第$(k-1)$步消元，在进行第 k 步消元之前，选出第 k 列中位于对角线及其以下元素绝对值中的最大者，即确定 t，使得：

$$\left|a_{tk}^{(k)}\right| = \max_{k \le i \le n}\left|a_{ik}^{(k)}\right| \tag{7-9}$$

将 $\overline{A}^{(k)}$ 的第 t 行和第 k 行互相交换，则元素 $a_{ik}^{(k)}$ 为新的主元素 $a_{kk}^{(k)}$，其余元素也均以交换后的位置表示，然后再按 Gauss 消元法进行第 k 步消元。这种方法称为列主元 Gauss 消元法。此时消元因子满足：

$$|l_{ik}| = \left|\frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}\right| \le 1,\ i = k+1, k+2, \cdots, n \tag{7-10}$$

在一般情况下，该法能保证舍入误差不扩散，即这个方法基本是稳定的。列主元 Gauss 消元法的运算量除选主元及行交换外和 Gauss 消元法是相同的。根据列主元消元法的思想，作者编写了函数文件 CMGelim.m 来实现列主元消元法解线性方程组，其中调用了前面提到的 uelim.m 程序。

例 7-5：用列主元消元法求下面的方程。

$$\begin{cases}4x_1+2x_2-6x_3+4x_4=2\\3x_1+x_2+5x_3+2x_4=0\\2x_1+3x_2+2x_3-x_4=0\\2x_1+4x_2+x_3+2x_4=8\end{cases}$$

程序如下：

```
A = [4,2,-6,4;3,1,5,2;2,3,2,-1;2,4,1,2];  % 输入系数矩阵 U
b = [2;0;0;8];                             % 输入向量 B
X = CMGelim(A,b)                          % 列主元 Gauss 消元法解方程组
```

输出结果为：

```
X =   -2.3254    2.1124    0.1657    2.0178
```



如果在列主元消元法的程序 CMGelim.m 中注释掉关于挑选主元的程序段（第 12~15 行），则可得到 Gauss 消元法的程序。

7.4 矩阵分解算法

利用矩阵的 LU，QR 和 cholesky 分解求线性方程组的解，在求解大型方程组的时候很有用。其优点是运算速度快、节省磁盘空间、节省内存。

从前一节的消元过程可以看出，Gauss 消元过程实际就是对方程组 AX=B 的增广矩阵[A,B]连续左乘以 $L_1,L_2,\ldots,L_{n-1}$ 得到上三角方程组 $UX=y$（y 为转化的增广矩阵 $\overline{A}^{(n)}$ 最右侧一列），即：

$$L_{n-1}L_{n-2}\cdots L_1AX=UX=y=L_{n-1}L_{n-2}\cdots L_1B \tag{7-11}$$

其中，

$$L_k=\begin{bmatrix}1 & & & & & & & \\ 0 & 1 & & & & & & \\ \vdots & \vdots & \ddots & & & & & \\ 0 & 0 & \cdots & 1 & & & & \\ 0 & 0 & \cdots & -l_{k+1,k} & 1 & & & \\ 0 & 0 & \cdots & -l_{k+2,k} & 0 & 1 & & \\ \vdots & \vdots & \cdots & \vdots & & & \ddots & \\ 0 & 0 & \cdots & -l_{nk} & 0 & \cdots & & 1\end{bmatrix} \tag{7-12}$$

令 $L=L_1^{-1}L_2^{-1}\cdots L_{n-1}^{-1}$，用 L 左乘公式（7-11），有：

$$LL_{n-1}L_{n-2}\cdots L_1AX = UX \tag{7-13}$$

于是得到：

$$A = LU \tag{7-14}$$

称 L 为单位下三角矩阵，U 为上三角矩阵。我们把这种矩阵 A 分解为两个或者多个简单矩阵的乘积称为矩阵分解。这里分解式（7-14）称为矩阵的 LU 分解。

例 7-6：用 LU 分解法解下面的方程。

$$\begin{cases} 3x_1 + 2x_2 + 2x_3 + 5x_4 = 2 \\ 2x_1 + 2x_2 + 3x_3 + 5x_4 = 0 \\ 2x_1 + 3x_2 + 4x_3 + 2x_4 = 0 \\ 2x_1 - 3x_2 + 3x_3 + 2x_4 = 9 \end{cases}$$

求解程序如下：

```
A = [3,2,2,5;2,2,3,5;2,3,4,2;2,-3,3,2];  % 输入系数矩阵 A
B = [2;0;0;9];                          % 输入向量 B
[L,U] = lu(A);                         % 计算 LU 分解
B1 = L\B;                             % 更新向量 B
X = uelim(U,B1)                        % 用消元法解上三角方程组
Delta = [A*X'-B]'                       % 检验结果
```

输出结果为：

```
X =    2.2477   -1.5413    0.2477   -0.4312
Delta =  1.0e-015 *
   -0.4441    0.4441   -0.4441         0
```

可见误差很小，在 10^{-16} 量级。

对矩阵 X 进行 QR 分解，就是把 X 分解为一个正交矩阵 Q 和一个上三角矩阵 R 的乘积形式。MATLAB 提供的函数 qr 可用于对矩阵进行 QR 分解，进行 QR 分解后，线性方程组 AX=B 的解可以通过下面的表达式语句计算：

```
X = uelim(U,Q\B)  % 用消元法解上三角方程组
```

例 7-7：用 QR 分解求解下面的线性方程组。

$$\begin{cases} 2x_1 + 2x_2 - 5x_3 + 3x_4 = 8 \\ 3x_1 - 5x_2 + 2x_3 + 7x_4 = 0 \\ 4x_1 + 2x_2 + 2x_3 - 5x_4 = 4 \\ 3x_1 + 3x_2 - 3x_3 - 5x_4 = 2 \end{cases}$$

求解程序如下：

```
A = [2,2,-5,3;3,-5,2,7;4,2,2,-5;3,3,-3,-5];     % 输入系数矩阵 A
B = [8;0;4;2];                               % 输入向量 B
[Q,R] = qr(A);                              % 计算 LU 分解
```

```
X = uelim(R,Q\B)                          % 用消元法解上三角方程组
```

计算结果如下：

```
X =    0.9489    2.8786    0.7859    1.4249
```

采用 QR 分解求解方程组还存在第 2 种格式，具体过程如下：

```
[Q,R,E] = qr(A);
X = E*(R\(Q\B))
```

当方程组的系数矩阵是对称正定时，可以直接做 Gauss 消元法，即对称正定矩阵保证能直接做 LU 分解。现在来研究此时 L 和 U 间的关系。因为矩阵 A 是对称的，故有：

$$a_{ij}=a_{ji},\ i,j=1,2,\cdots,n \tag{7-15}$$

由 LU 分解分式：

$$u_{1i}=\bar{a}_{1i},i=1,2,\cdots,n,n+1 \text{，（}\bar{a}_{i,j}\text{ 表示增广矩阵的元素）} \tag{7-16}$$

$$l_{i1}=\frac{a_{i1}}{a_{11}},\ i=2,3,\cdots,n \tag{7-17}$$

由上述公式（7-16）和（7-17）可得：

$$l_{i1}=\frac{a_{1i}}{a_{11}}=\frac{u_{1i}}{a_{11}},\ i=2,3,\cdots,n \tag{7-18}$$

若已求得第 1 步到第$(k\text{-}1)$步，则矩阵 L 和 U 的元素有如下关系：

$$l_{ij}=\frac{u_{ji}}{u_{jj}},j=1,2,\cdots,k-1,i=j+1,\cdots,n \tag{7-19}$$

由此可知，对一切 k 均有：

$$l_{ik}=\frac{u_{ki}}{u_{kj}},\ k=1,2,\cdots,n-1,i=k+1,k+2,\cdots,n \tag{7-20}$$

对于对称矩阵，按公式（7-20）求解，而不直接采用上面的矩阵分解法，计算量将会减少近一半。而且上述分解也是唯一的，其中 $\mathrm{L}'=\mathrm{U}$。这种方法通常也称为改进平方根法或 cholesky 分解法。矩阵 L 元素的表达式可以写为下面的递推形式：

$$\begin{cases} l_{1,1}=\sqrt{a_{1,1}},\ l_{i,1}=a_{i,1}/l_{1,1},\ i=2,3,\dots,n \\ l_{i,i}=\sqrt{a_{i,i}-\sum\limits_{k=1}^{r-1}l_{i,k}},i=2,3,\dots,n \\ l_{i,r}=\left(a_{i,r}-\sum\limits_{k=1}^{r-1}l_{i,k}l_{r,k}\right)\Big/l_{r,r},i=r+1,\dots,n \end{cases} \tag{7-21}$$

作者根据公式（7-21）编写了 Cholesky.m 文件（该文件保存在光盘中）实现 cholesky 分解，该函数在进行 cholesky 分解前，先检查矩阵 A 是否为对称正定，如果不是，则程序将不再对矩阵

A 进行 cholesky 分解。

例 7–8： 利用 cholesky 分解求下面的方程。

$$\begin{bmatrix}8 & 2 & 4 & 1 & 4\\ 2 & 9 & 8 & 2 & 3\\ 4 & 8 & 9 & 3 & 2\\ 1 & 2 & 3 & 7 & 1\\ 4 & 1 & 2 & 3 & 7\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\\ x_4\\ x_5\end{bmatrix}=\begin{bmatrix}5\\ 4\\ 3\\ 2\\ 1\end{bmatrix}$$

求解程序如下：

```
A = [8,2,4,1,4;2,9,8,2,3;4,8,9,3,2;1,2,3,7,1;4,3,2,1,7];    % 输入系数矩阵 A
B = [5;4;3;2;1];                                          % 输入向量 B
L = Cholesky(A);                                          % cholesky 分解
X = uelim(L',L\B)                                         % 用消元法解上三角方程组
Delta = [A*X'-B]'                                         % 计算误差
```

输出为：

```
X =    3.3736    5.5956   -5.9253    1.1231   -2.6505
Delta =  1.0e-014 *
0.1776   -0.5329    0.6217    0.2220    0.7105
```

下三角矩阵 L 等于：

```
L =
    2.8284         0         0         0         0
    0.7071    2.9155         0         0         0
    1.4142    2.4010    1.1114         0         0
    0.3536    0.6002    0.9527    2.3679         0
    1.4142    0.6860   -1.4819    0.6335    1.3900
```

计算 L*L'，则有：

```
ans =    8.0000    2.0000    4.0000    1.0000    4.0000
         2.0000    9.0000    8.0000    2.0000    3.0000
         4.0000    8.0000    9.0000    3.0000    2.0000
         1.0000    2.0000    3.0000    7.0000    1.0000
         4.0000    3.0000    2.0000    1.0000    7.0000
```

可见关系 $A = LL'$ 是成立的。

作为比较，可以对本例中的矩阵 ***A*** 进行 LU 分解，结果如下：

```
[L,U] = lu(A)
L =    1.0000         0         0         0         0
       0.2500    1.0000         0         0         0
       0.5000    0.8235   -0.7500    0.1712    1.0000
       0.1250    0.2059   -0.6429    1.0000         0
       0.5000    0.2353    1.0000         0         0
U =    8.0000    2.0000    4.0000    1.0000    4.0000
            0    8.5000    7.0000    1.7500    2.0000
            0         0   -1.6471    0.0882    4.5294
```

```
     0         0          0    6.5714    3.0000
     0         0          0         0    1.2364
```

对比上面的程序可以看出，上面的结果与我们自行编写的 cholesky 程序计算得到的下三角矩阵 $\boldsymbol{L}$ 是不同的。

如果矩阵 $\boldsymbol{A}$ 或者常数项 B 的微小变化可以引起方程组 AX=B 解的巨大变化，则称此方程组为"病态"方程组，同时称矩阵 $\boldsymbol{A}$ 为"病态"矩阵（这里是相对于方程组而言）。反之则称方程组为"良态"方程组，相应地 $\boldsymbol{A}$ 为"良态"矩阵。矩阵的"病态"性质是矩阵本身的特性。为了定量地描述方程组的"病态"程度，下面对方程组 AX=B 的系数矩阵和右端常数项分别进行两种情形的分析。

首先考虑线性方程组 AX=B 的系数矩阵 $\boldsymbol{A}$ 的扰动。假设此时常数项 B 不发生扰动，记系数矩阵 $\boldsymbol{A}$ 的扰动为 δA，相应解的扰动记为 δx，那么 $(A+\delta A)(x+\delta x)=b$，即：

$$A\delta x+\delta A(x+\delta x)=0 \tag{7-22}$$

对公式（7-22）两边同时取范数：

$$\|\delta x\|=\left\|A^{-1}\delta A(x+\delta x)\right\|\le\left\|A^{-1}\right\|(\|x\|+\|\delta x\|) \tag{7-23}$$

公式（7-23）中 $\|\cdots\|$ 表示求范数。如果 δA 足够小，使得 $\left\|A^{-1}\right\|\|\delta A\|<1$ 成立，则由上式可以得到：

$$\frac{\|\delta x\|}{\|x\|}\le\frac{\left\|A^{-1}\right\|\|\delta A\|}{1-\left\|A^{-1}\right\|\|\delta A\|}=\frac{\|A\|\left\|A^{-1}\right\|\frac{\|\delta A\|}{\|A\|}}{1-\|A\|\left\|A^{-1}\right\|\frac{\|\delta A\|}{\|A\|}} \tag{7-24}$$

上式表明当系数矩阵 $\boldsymbol{A}$ 有扰动时，解的扰动和 $\|A\|\left\|A^{-1}\right\|$ 有关。一般地，$\|A\|\left\|A^{-1}\right\|$ 越大，解的扰动也越大。

接下来考虑常数项 B 的扰动，记为 δB，相应解 x 的扰动仍记为 δx，那么 $A(x+\delta x)=B+\delta B$，即 $A\delta x=\delta B$，或者 $\delta x=A\delta B$。

两边取范数有 $\|\delta x\|=\left\|A^{-1}\delta B\right\|\le\left\|A^{-1}\right\|\|\delta B\|$，因为 $\|x\|\ge\frac{\|Ax\|}{\|A\|}=\frac{\|B\|}{\|A\|}$，所以：

$$\frac{\|\delta x\|}{\|x\|}\le\frac{\left\|A^{-1}\right\|\|\delta B\|}{\frac{\|B\|}{\|A\|}}=\|A\|\left\|A^{-1}\right\|\frac{\|\delta B\|}{\|B\|} \tag{7-25}$$

式（7-25）表明当右侧常数项 B 有扰动时，解的误差不超过 B 的相对误差的 $\|A\|\left\|A^{-1}\right\|$ 倍数。

综合上面两种情况可知，$\|A\|\|A^{-1}\|$ 实际上决定了解对原始数据变化的灵敏程度，也就是说描述了方程组的“病态”程度，因此把 $\|A\|\|A^{-1}\|$ 定义为矩阵 $\boldsymbol{A}$ 的条件数，即：

$$\text{cond}(A)_s = \|A\|_s \|A^{-1}\|_s \text{，其中 } s = 1, 2, \infty \tag{7-26}$$

在条件数表达式中，可以选择 1，2 和 ∞ 范数。其中取 2 和 ∞ 范数对应的条件数比较常用，特别情况下，当取 2- 范数时，条件数也称为“谱条件数”，即：$\text{cond}(A)_2 = \|A\|_2 \|A^{-1}\|_2 = \sqrt{\dfrac{\lambda_{\max}(A'A)}{\lambda_{\min}(A'A)}}$。当 $\boldsymbol{A}$ 为对称矩阵时，$\text{cond}(A)_2 = \lambda_{\max}/\lambda_{\min}$，其中 $\lambda_{\max}$ 和 $\lambda_{\min}$ 分别是矩阵 $\boldsymbol{A}$ 的绝对值最大和最小本征值。

对于非奇异矩阵 $\boldsymbol{A}$，条件数具有如下 4 条性质：

- $\text{cond}(A)_s \geq 1$。
- $\text{cond}(cA)_s = \text{cond}(A)_s$。
- 若 $\boldsymbol{A}$ 是正交矩阵，$\text{cond}(A)_2 = 1$。
- 若 $\boldsymbol{R}$ 为正交矩阵，$\text{cond}(RA)_2 = \text{cond}(AR)_2 = \text{cond}(A)_2$。

在 MATLAB 中提供了函数 cond 来计算条件数，其调用格式为：

```
C = cond(X, p);
```

参数说明： C 是输出的条件数。X 是输入的矩阵。p 用来指定哪一种范数，取值可以是 1，2，inf 和 fro，p 的默认值是 2。

希尔伯特（Hilbert）矩阵定义为 $H_{i,j} = 1/(i+j-1)$。在 MATLAB 中函数 hilb 可以给出 n 阶希尔伯特矩阵。调用函数 cond 计算相应的 3，5 和 6 阶希尔伯特矩阵的条件数有：

```
c3 = cond(hilb(3),inf)
c5 = cond(hilb(5),inf)
c6 = cond(hilb(6),inf)
```

输出如下：

```
c3 =  748.0000
c5 =  9.4366e+005
c6 =  2.9070e+007
```

可见随着 n 的增大，希尔伯特矩阵的条件数增大，也就是说“病态”越严重。

设计 X=[1;1;1]为方程组 Hilb(3)*X=B 的解，因此可知 B=[11/6;13/12;47/60]。这里考虑 3 阶希尔伯特矩阵以及常数项 B 发生微小的扰动，即 $\begin{bmatrix} 1.00 & 0.500 & 0.333 \\ 0.500 & 0.333 & 0.250 \\ 0.333 & 0.250 & 0.200 \end{bmatrix} \begin{bmatrix} x_1 + \delta x_1 \\ x_2 + \delta x_2 \\ x_3 + \delta x_3 \end{bmatrix} = \begin{bmatrix} 1.83 \\ 1.08 \\ 0.784 \end{bmatrix}$，简记为

$(H_3+\delta H_3)(x+\delta x)=B+\delta B$，其中 $x=\begin{bmatrix}1\\1\\1\end{bmatrix}$。

利用下面的语句可以求解上面的方程组：

```
A = [1,0.5,0.333;0.5,0.333,0.25;0.333,0.25,0.2];
B = [1.83;1.08;0.784];
X = [A\B]'
```

输出结果如下：

```
X =1.1228    0.2910    1.6868
```

对比可以得到 $\delta x=[-0.0562,\ 0.3151,\ -0.2952]'$，计算相对误差有 $\frac{\|\delta H_3\|_\infty}{\|H_3\|_\infty}\approx 1.82\times10^{-4}<0.02\%$，$\frac{\|\delta B\|_\infty}{\|B\|_\infty}\approx 1.8\times10^{-3}<0.2\%$，$\frac{\|\delta B\|_\infty}{\|B\|_\infty}\approx 0.709=70.9\%$。

可见 H_3 和 B 的相对误差不超过 0.2%，而解 X 变化的相对误差达到 70%。

计算条件数需要得出矩阵的逆，因此有的时候可能比较困难。根据数值经验，在下面的 4 种情况下，方程组一般是“病态”的。

- 在用列主元消元法时出现小主元。
- 如果矩阵 A 的最大本征值和最小本征值之比（取绝对值）是大的，则矩阵 A 是“病态”的。
- 系数矩阵中有行（或列）近似线形相关，或者系数行列式的值接近于 0。这种情况只是经验判断，并不是绝对的。如当 $A=\varepsilon I$ 时，ε 是很小的数，矩阵 A 的行列式 det(A)会以 ε^n 趋近于 0，但是矩阵 A 的条件数是等于 1，方程组的状态良好。
- 系数矩阵 A 的元素间数量级相差很大，并且无一定规则，那么矩阵 A 可能是“病态”的。

用列主元消元法不能解决病态方程求解问题。对病态方程组求解可采用以下方法：

- 采用高精度运算，减轻病态影响，例如用双倍字长运算。
- 用预处理方法改善 A 的条件数，即选择非奇矩阵 P，Q，使 $PAQ(Q^{-1}x)=PB$ 与 AX=B 等价，而 $\tilde{A}=PAQ$ 的条件数较矩阵 A 有所改善，则求 $\tilde{A}\tilde{x}=\tilde{B}=PB$ 的解，即 $\hat{x}=Q^{-1}x$ 为原方程的解。
- 平衡方法，当 A 中元素的数量级相差很大，可以采用行均衡或列均衡的方法改善 A 的条件数。设 A 为非奇异，计算 $s_i=\max\limits_{1\le j\le n}|a_{i,j}|(i=1,2,\cdots,n)$，令 $D=\mathrm{diag}(1/s_1,1/s_2,\cdots,1/s_n)$，于是求 AX=B 等价于求 $DAx=DB$，或 $\tilde{A}x=\tilde{B}$，这时 $\tilde{A}=DA$ 的条件数可得到改善，这就是行均衡法。

奇异值分解（sigular value decomposition）是另一种正交矩阵分解法，它是最可靠的分解法，但是它比 QR 分解法要花上近十倍的计算时间。MATLAB 中提供了函数 svd 来计算奇异值分解。其调用格式如下：

```
[U, S, V] = svd(A);
```

其中 U 和 V 代表两个相互正交矩阵，而 S 代表一对角矩阵。和 QR 分解法相同，A 可以不是方矩阵。使用奇异值分解法的用途是解最小平方误差法和数据压缩。用这个分解来计算图形信息，性能相当好。奇异值分解法可用于求解线性方程组。在计算线性方程组时，一些不能分解的矩阵或者严重病态矩阵的线性方程都能得到解。

7.5 迭代法

解线性方程组 AX=B 的迭代法是从初始解出发，根据设计好的步骤用逐次求出的近似解逼近精确解。前面介绍的解线性方程组的直接方法一般适合于 $\boldsymbol{A}$ 为低阶稠密矩阵（指 n 不大且元多为非零）的情况，而在工程技术和科学计算中常会遇到大型稀疏矩阵（指 n 很大且零元较多）的方程组，迭代法在计算和存储两方面都适合后一种情况。由于迭代法是通过逐次迭代来逼近方程组的解的，所以收敛性和收敛速度是构造迭代法时应该考虑的问题。因为不同的系数矩阵具有不同的性态，所以大多数迭代方法都具有一定的适用范围。有时某种方法对于一类方程组迭代收敛，而对另一类方程组迭代时就发散。

首先介绍雅可比（Jacobi）迭代法解线性方程组 AX=B。对于线性方程组 AX=B，矩阵 $\boldsymbol{A}$ 非奇异，把矩阵 $\boldsymbol{A}$ 分解为 $A=D-L-U$ 。其中矩阵 $D=\mathrm{diag}(A)$，$\boldsymbol{L}$ 和 $\boldsymbol{U}$ 分别是矩阵 $\boldsymbol{A}$ 的严格下三角部分和上三角部分，即：

$$L=\begin{bmatrix} 0 & & & \\ a_{21} & 0 & & \\ \vdots & \vdots & \ddots & \\ a_{n1} & \cdots & a_{n,n-1} & 0 \end{bmatrix},\quad U=\begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ & 0 & \cdots & \vdots \\ & & \ddots & a_{n-1,n} \\ & & & 0 \end{bmatrix} \tag{7-27}$$

迭代公式为 $x_{k+1}=D^{-1}(L+U)x_k+D^{-1}B$，当我们记 $\tilde{A}=D^{-1}(L+U)$，$\tilde{B}=D^{-1}B$，那么具体的 $\tilde{A}$ 和 $\tilde{B}$ 可以表示为：

$$\tilde{A}=\begin{bmatrix} 0 & -a_{12}/a_{11} & \cdots & -a_{1n}/a_{11} \\ -a_{21}/a_{22} & 0 & \cdots & -a_{2n}/a_{22} \\ \vdots & \vdots & \ddots & \vdots \\ -a_{n1}/a_{nn} & -a_{n2}/a_{nn} & \cdots & 0 \end{bmatrix},\quad \tilde{B}=\begin{bmatrix} b_1/a_{11} \\ b_2/a_{22} \\ \vdots \\ b_n/a_{nn} \end{bmatrix} \tag{7-28}$$

雅可比迭代法的收敛充要条件是矩阵 $\tilde{A}$ 的谱半径小于 1，即 $\rho(\tilde{A})<1$（$\rho(\tilde{A})=\lim\limits_{i}|\lambda_i|$，其中 λ_i 是矩阵 $\tilde{A}$ 的本征值），谱半径在 MATLAB 中可以使用语句 max(abs(eig($\tilde{A}$)))来计算。

根据公式（7-28），编写 Jacobi_iteration.m 实现其描述的雅可比迭代法求解线性方程组。其调用格式为：

```
X = Jacobi_iteration(A,B,X0,kmax,tol);
```

参数说明： X 是返回的解。A 是系数矩阵。B 是常数列向量。X0 是迭代初始值，默认值为 zeros(size(B))。Kmax 指定最大迭代次数，默认值为 500。tol 是预先指定的精度，默认值是 1e-6。

例 7-9： 利用雅可比迭代法求解下面的线性方程组。

$$\begin{bmatrix}9 & 4 & 2\\3 & 8 & 2\\2 & 3 & 7\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix}=\begin{bmatrix}1\\2\\3\end{bmatrix}$$

求解程序如下：

```
A = [9,4,2;3,8,2;2,3,7];                    % 输入系数矩阵 A
B = [1;2;3];                            % 输入常数项列向量 B
X0 = [0,0,0];                           % 初始值
X = Jacobi_iteration(A,B,X0)                % 用雅可比迭代法解线性方程组
```

输出如下：

```
X =   -0.0489    0.1766    0.3668
```

雅可比迭代法还存在一种扩展形式，即：

$$\begin{aligned}x^{(k+1)} &= x^{(k)} - \omega D^{-1}\left[Ax^{(k)} - B\right]\\ &= x^{(k)} - \omega D^{-1}Ax^{(k)} + \omega D^{-1}B\\ &= \left(I_n - \omega D^{-1}A\right)x^{(k)} + \omega D^{-1}B\end{aligned} \tag{7-29}$$

其中矩阵 $D=\mathrm{diag}(A)$。这种迭代法被称为雅可比超松弛法（Jacobi over relaxation），简称 JOR 法。当雅可比迭代收敛时，JOR 法对于 $0<\omega\le 1$ 收敛。

在公式（7-29）中去掉 D^{-1} 可以得到 RF 法（Richardson method）的迭代公式：

$$x^{(k+1)} = \left(I_n - \omega A\right)x^{(k)} + \omega B \tag{7-30}$$

其中 $I_n-\omega A$ 被称为 RF 法的迭代矩阵。进一步可以使用一个对角矩阵 Ω 代替公式（7-30）中的参数 ω 而得到广义理查森法，简称为 GRF 法。在雅可比迭代法程序的基础上可以根据公式(7-29）和（7-30）分别得到 RF 法和 GRF 法的计算程序。

在雅可比迭代法中，矩阵 $x_{k+1}=\tilde{A}x_k+\tilde{B}$ 控制的迭代过程使第 $k+1$ 步的迭代需要用到前面第 k 步中计算得到的 $\{x_k\}$，这样各点的收敛情况完全取决于迭代方程 $x_{k+1}=\tilde{A}x_k+\tilde{B}$，各点之间的影响没有考虑进去。下面来介绍高斯-赛德尔（Gauss-Seidel）迭代法。

把矩阵 $\boldsymbol{A}$ 分解为 $\boldsymbol{A}=\boldsymbol{D}+\boldsymbol{L}+\boldsymbol{U}$，其中 $\boldsymbol{D}$，$\boldsymbol{L}$ 和 $\boldsymbol{U}$ 的意义相同于雅可比迭代法中的意义。高斯-赛德尔迭代法的矩阵形式如下：

$$x^{(k+1)} = (D-L)^{-1}Ux^{(k)} + (D-L)^{-1}B \tag{7-31}$$

其中$(D-L)^{-1}U$被称为高斯-赛德尔迭代法的迭代矩阵，记为M_{GS}。在这个方法中，考虑了x各分量的互相影响，把公式（7-31）写为如下分量形式：

$$x_m^{(k+1)}=\frac{b_m-\sum_{p=1}^{m-1}a_{mp}x_p^{(k+1)}-\sum_{p=m+1}^{n}a_{mp}x_p^{(k)}}{a_{mm}} \tag{7-32}$$

从高斯-赛德尔迭代法中我们可以看出，迭代式的构造和雅可比迭代式的构造整体上是类似的，只不过在计算x_k的第$k+1$步时，用到了第$k+1$步前已经得到的$\{x_1^{(k+1)},x_2^{(k+1)},\dots,x_{m-1}^{(k+1)}\}$，而后面未计算的分量$\{x_{m+1},x_{m+2},\dots,x_n\}$用第$k$步结果$\{x_{m+1}^{(k)},x_{m+2}^{(k)},\dots,x_n^{(k)}\}$代替。高斯-赛德尔迭代法收敛的充要条件是迭代矩阵$M_{GS}$的谱半径小于1，即$\rho(M_{GS})<1$。此外，当矩阵$A$对称正定时，用高斯-赛德尔迭代法解方程组AX=B时迭代过程收敛。在保证雅可比迭代法和高斯-赛德尔迭代法都收敛的前提下，高斯-赛德尔迭代法的收敛速度要比雅可比迭代法快。

作者根据高斯-赛德尔迭代公式编写了相应的程序gauseid.m（完整的函数文件保存在光盘中），该函数调用格式如下：

```
X = gauseid(A,B,X0,kmax,tol);
```

参数说明： X 是返回的解。A 是系数矩阵。B 是常数列向量。X0 是迭代初始值，默认值为zeros(size(B))。Kmax 用于设定最大迭代次数，默认值为 500。tol 是预先指定的精度，默认值是1e-6。该函数在没有输出的时候将显示每一步迭代的结果。

例 7-10： 利用高斯-赛德尔迭代法求下面方程组的解。

$$\begin{bmatrix}9 & 2 & 2 & -4\\2 & 8 & 2 & -3\\2 & 2 & 7 & 2\\3 & 2 & -2 & 9\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\\x_4\end{bmatrix}=\begin{bmatrix}1\\-1\\2\\1\end{bmatrix}$$

计算程序如下：

```
A = [9,2,2,-4;2,8,2,-3;2,2,7,2;3,2,-2,9];    % 输入系数矩阵 A
B = [1;-1;2;1];                              % 输入列向量 B
X0 = [0,0,0,1];                              % 输入迭代初值
X = gauseid(A,B,X0)                          % 利用高斯-赛德尔迭代计算方程组的根
D = [A*X'-B]'                                % 计算误差
```

结果如下：

```
X =    0.1610   -0.1704    0.2456    0.1499
D =  1.0e-006 *
  -0.8279   -0.4502    0.0925         0
```

可见达到了默认精度时程序停止。

此外雅可比迭代法和高斯-赛德尔迭代法的思想还可以用来计算非线性方程的解，下面考虑一个非线性方程求解的例子。

例 7-11： 计算下面非线性方程的根。

$$\begin{cases} x^2+9x+3y^2-12=0 \\ 2x^3+6y-y^2-6=0 \end{cases}$$

首先把方程组转化为下面的形式便于迭代。

$$\begin{cases} x=\left(12-x^2-3y^2\right)/9 \\ y=\left(6-2x^3+y^2\right)/6 \end{cases}$$

进一步得到下面的迭代形式。

雅可比迭代：$\begin{cases} x_{k+1}=\left(12-x_k^2-3y_k^2\right)/9 \\ y_{k+1}=\left(6-2x_k^2+y_k^2\right)/6 \end{cases}$

高斯-赛德尔迭代：$\begin{cases} x_{k+1}=\left(12-x_k^2-3y_k^2\right)/9 \\ y_{k+1}=\left(6-2x_{k+1}^2+y_k^2\right)/6 \end{cases}$

根据上面的迭代式进行计算。

下面的程序是根据公式（7-28）编写的雅可比迭代程序。

```
kmax = 500;               % 预定的迭代次数
tol = 1e-6;                 % 预定的精度
x = [0,0];                  % 迭代初值
for k = 1:kmax
   xp = x;                                % 记录上一次迭代结果
   x(1) = (12-xp(1)^2-2*xp(2)^2)/9;        % 对 x 进行迭代
   x(2) = (6-2*xp(1)^3+xp(2)^2)/6;         % 对 y 进行迭代
   if norm(x-xp)/(norm(xp)+eps)<tol;       % 检查精度
      break;                             % 满足条件结束循环
   end
end
k, x                                     % 输出迭代次数和求解结果
```

输出如下：

```
k =    32
x =1.1161    0.5958
```

下面的程序是根据公式（7-32）编写的高斯-赛德尔迭代程序。

```
% 利用高斯-赛德尔迭代求解非线性方程
kmax = 500;               % 预定的迭代次数
tol = 1e-6;                 % 预定的精度
x = [0,0];                  % 迭代初值
for k = 1:kmax
   xp = x;                                     % 记录上一次迭代结果
   x(1) = (12-x(1)^2-2*x(2)^2)/9;                     % 对 x 进行迭代
   x(2) = (6-2*x(1)^3+x(2)^2)/6;                    % 对 y 进行迭代
   if norm(x-xp)/(norm(xp)+eps)<tol;                 % 检查精度
```

```
        break;                                    % 满足条件结束循环
    end
end
k, x                                              % 输出迭代次数和求解结果
```

输出结果如下：

```
k =     15
x =     1.1161    0.5958
```

可见高斯-赛德尔迭代的速度比雅可比迭代快很多。逐次超松弛法（Successive over relaxation method）是求解大型稀疏矩阵方程组的有效方法，简称为 SOR 法。其分量迭代公式如下：

$$x_i^{(k+1)}=(1-\omega)x_i^{(k)}+\omega\left(\frac{b_i}{a_{ii}}-\sum_{j=1}^{i-1}\frac{a_{ij}}{a_{ii}}x_j^{(k+1)}-\sum_{j=i+1}^{n}\frac{a_{ij}}{a_{ii}}x_j^{(k)}\right)\quad (i=1,2,\ldots,n) \tag{7-33}$$

与高斯-赛德尔迭代法的分量公式相比，逐次超松弛法的迭代公式中增加了参数 ω（其被称为松弛参数或者松弛因子，relaxation factor）。当 $0<\omega<1$ 时，公式（7-33）描述的是低松弛方法（under-relaxation method），对于一些方程组，使用高斯-赛德尔迭代法得不到收敛解或者不收敛，使用低松弛方法却可能得到收敛解。当 $\omega>1$ 时，公式（7-33）描述的是超松弛方法（over relaxation method），此时可以加速高斯-赛德尔迭代方法的收敛。在 $\omega=1$ 的时候，公式（7-33）等同于公式（7-32）。公式（7-33）的矩阵形式如下：

$$x^{(k+1)}=(D-\omega L)^{-1}\left[(1-\omega)D+\omega U\right]x^{(k)}+\omega(D-\omega L)^{-1}B$$
$$M_{SOR}=(D-\omega L)^{-1}\left[(1-\omega)D+\omega U\right] \tag{7-34}$$

其中 M_{SOR} 被称为逐次超松弛法的迭代矩阵，矩阵 D，L 和 U 的意义同雅可比迭代法。对矩阵 A 的对角元素都是非零实数时，对于迭代矩阵 M_{SOR} 的谱半径存在关系 $\rho(M_{SOR})\ge|\omega-1|$。逐次超松弛法收敛的充要条件是迭代矩阵 M_{SOR} 的谱半径小于 1，即 $\rho(M_{SOR})<1$。

7.6 共轭梯度法解方程组

共轭梯度法（Conjugate gradient method）简称 CG 法，它是求解系数矩阵为对称正定大型稀疏方程组的一种有效方法之一。该方法的理论基础是变分原理，即通过计算 n 元二次函数 $\varphi(x)$ 的极小值点 $\tilde{x}$ 得到方程组 AX=B 的近似解。这里函数 $\varphi(x)$ 可以表示为：

$$\varphi(x)=\frac{1}{2}(x,Ax)-(x,B) \tag{7-35}$$

其中 (P,Q) 表示向量的内积，即：$(P,Q)=\sum_i P_iQ_i$。

从而求解方程组 AX=B 的解转化为求解函数 $\varphi(x)$ 的极小值问题。MATLAB 提供与共轭梯度法相关的函数如表 7.2 所示。

表 7.2　与共轭梯度法相关的函数

函数名	说明	函数名	说明
bicg	双共轭梯度法	minres	最小残差法
bicgstab	稳定双共轭梯度法	pcg	预处理共轭梯度法
cgs	复共轭梯度平方法	qmr	准最小残差法
gmres	广义极小残差法	symmlq	对称 LQ 法
lsqr	共轭梯度的 LSQR 方法		



这些函数的用法类似，下面以函数 bicg，bicgstab，cgs 和 lsqr 为例进行说明。

MATLAB 提供的函数 bicg 可以实现双共轭梯度法求解方程组，其调用格式为：

```
x = bicg(A, B);
x = bicg(A, B, tol);
x = bicg(A, B, tol, maxit);
x = bicg(A, B, tol, maxit, M);
x = bicg(A, B, tol, maxit, M1, M2);
x = bicg(A, B, tol, maxit, M1, M2, x0);
[x, flag] = bicg(A, B);
[x, flag, relres] = bicg(A, B);
[x, flag, relres, iter] = bicg(A, B);
[x, flag, relres, iter, resvec] = bicg(A, B);
```

参数说明： x 是线性方程组 AX=b 的解。flag 的取值有 5 种可能：0 表示在指定迭代次数之内按要求精度收敛；1 表示在指定迭代次数内不收敛；2 表示 M 为坏条件的预处理因子；3 表示两次连续迭代完全相同；4 表示标量参数太小或太大。relres 表示相对误差 norm(b-A*x)/norm(b)。iter 表示计算的迭代次数。resvec 表示每次迭代的残差，即 norm(b-A*x0)。矩阵 A 必须为 n 阶方阵，B 为常数项列向量。A 也可以是由函数文件定义并返回 A*X 的函数。tol 是指定的误差，其默认值是 1e-6。maxit 是最大迭代次数。M 为用于对称正定矩阵的预处理因子。参数 M，M1，M2 之间的关系是 M=M1×M2。x0 为初始估计值，默认值为全 0 列向量。函数 bicg 在运行后，如果收敛，将显示结果信息；如果收敛失败，将给出警告信息并显示相对残差 norm(b-A*x)/norm(b)和计算终止的迭代次数。

例 7-12： 利用双共轭梯度法求 AX=B 线性方程组，其中 A=gallery('dorr',120)，B=[ones(60,1); 3*ones(60,1)]。

相应的计算程序如下：

```
A = gallery('dorr',120); %生成系数矩阵 A
B = [ones(60,1);3*ones(60,1)]; %将矩阵 A 的各行求和，构成一列向量
maxit = 300; % 指定最大迭代次数
[x, flag, relres, iter, resvec] = bicg(A, B,[], maxit);
                %用函数 bicg 求解
flag, relres, iter, % 显示部分输出结果
```

输出为：

```
flag =     0
relres =  5.8311e-008
iter =   121
```

为了进一步显示收敛过程，在图 7.1 中画出了相对残差和迭代次数之间的曲线。可见迭代 105 次之前收敛较慢，在后面的几次迭代速度突然加快。相应绘图程序在光盘中的 bicg_test.m 程序中有详细说明。

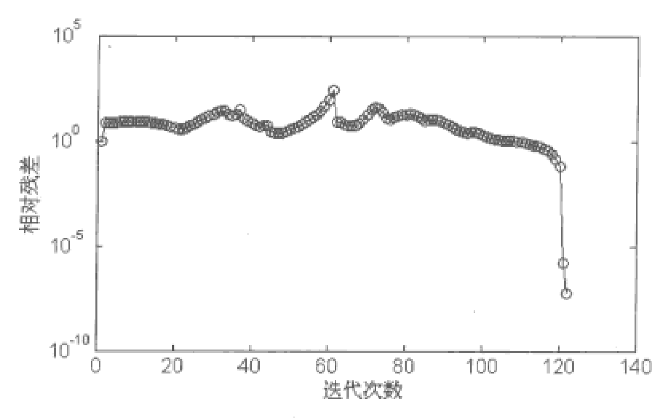


图 7.1　双共轭梯度法求解方程组的相对残差

MATLAB 提供函数 bicgstab 实现稳定双共轭梯度法解线性方程组，其调用格式为：

```
x = bicgstab(A, B);
x = bicgstab(A, B, tol);
x = bicgstab(A, B, tol, maxit);
x = bicgstab(A, B, tol, maxit, M);
x = bicgstab(A, B, tol, maxit, M1, M2);
x = bicgstab(A, B, tol, maxit, M1, M2, x0);
[x, flag] = bicgstab(A, B);
[x, flag, relres] = bicgstab(A, B);
[x, flag, relres, iter] = bicgstab(A, B);
[x, flag, relres, iter, resvec] = bicgstab(A, B);
```

函数 bicgstab 的参数意义与函数 bicg 相同，这里不再赘述。

这里使用 bicgstab 函数来求解前面例 7-11 中的方程，即把前面程序中的 bicg 换为 bicgstab 即可。所得结果的相对残差如图 7.2 所示，可见稳定双共轭梯度方法的收敛速度较慢。在使用函数 bicg 和 bicgstab 的时候，如果用函数 lu 或者 qr 生成矩阵 M1 和 M2，可以极大地增加收敛速度。

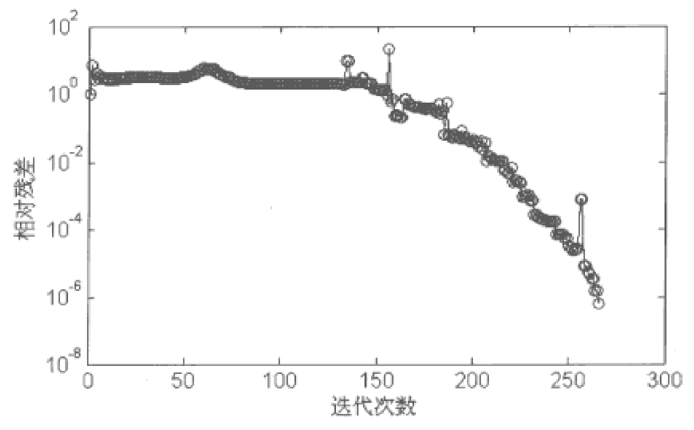


图 7.2　稳定双共轭梯度法求解方程组的相对残差

MATLAB 提供的函数 cgs 和 lsqr 可以分别实现复共轭梯度平方法和共轭梯度的 LSQR 方法求解线性方程组。它们的调用方法和函数 bicg 相同，读者可以参考前面的说明。比较使用函数 cgs 和 lsqr 求解例 7-12 的方程，所得残差结果如图 7.3 所示。图中在函数 cgs 迭代求解过程中，起伏现象比较明显；函数 lsqr 的计算中一直是收敛的，但是收敛得很慢。

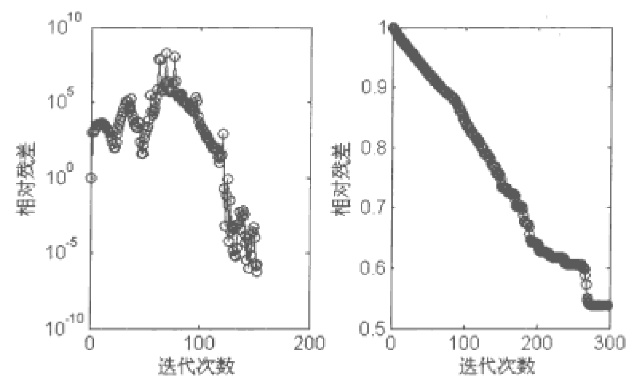


图 7.3　复共轭梯度平方法（左图）和共轭梯度的 LSQR 方法（右图）解线性方程组的相对残差图

7.7　小结

本章主要介绍了线性方程组的解法。首先介绍了一些矩阵操作的函数，包括矩阵的分解、矩阵参数的计算以及基本的矩阵操作，这些是后面求解线性方程组的基础。对于一般的线性方程组可以使用直接计算法，即通过矩阵求逆的操作得到方程组的解，其中对于正定方程组需要计算其通解。消元法可以逐一地得到方程组的解，本章给出了 Gauss 消元法、追赶法以及列主元消元法的介绍。在直接法当中，还介绍了通过矩阵的 LU 分解、QR 分解以及 cholesky 分解求解线性方程组的解。在迭代法的介绍中，主要给出了雅可比迭代法、高斯-赛德尔迭代法以及逐次松弛迭代法的详细介绍。最后简要介绍了共轭梯度法，以及相关的 MATLAB 函数。

◆　　◆　　◆

第 8 章 超越方程的求解

本章包括

- ◆ **函数解法** 包括利用函数 solve，fzero，fsolve 和 roots 计算方程的解。
- ◆ **间接求解** 转化为最小值问题，利用 fminbnd 函数求解。
- ◆ **多点初值迭代法** 介绍二分法和抛物线法。
- ◆ **牛顿法及演化版本** 介绍牛顿法、正割法和 Steffenson 法等求解方程。

在很多研究性问题中，我们常常遇到一些复杂的方程（组）。它们有的存在解析结果，但大多数的方程无法获得解析解（即超越方程）。通过计算机的程序计算，可以得到这些超越方程的足够高精度的数值解，能够满足我们的需要。本章将介绍一般方程和超越方程的解法，以及 MATLAB 提供的函数和一些数值方法。

8.1 函数解法

MATLAB 中提供了函数 solve，fzero，roots 以及 fminbnd 可以用来求解方程或者方程组。使用这些函数可以方便地求解不同形式的方程。本节将介绍这些函数的用法。

8.1.1 求解一般方程

使用 MATLAB 提供的函数 solve 可以方便地求解一般的方程，它的使用方法如下：

```
x = solve('eqn1','eqn2',...,'eqnN')
x = solve('eqn1','eqn2',...,'eqnN','var1,var2,...,varN')
x = solve('eqn1','eqn2',...,'eqnN','var1','var2',...'varN')
```

参数说明： x 是输出的计算结果，它是一个由符号型数据组成的结构体。eqn1，eqn2，…，eqnN 是符号表达式或者是方程对应的字符串，其中方程表达式是完整的，即 $f(x_1, x_2, ..., x_N)=0$ 的完整内容。var1，var2，…，varN 是符号变量或者是未知数的字符串。需要指出的是函数 solve 是按照英文字母表顺序把结果赋值给待求量的。

例 8–1： 利用 solve 函数求方程 $ax^2+bx+c=0$ 的解。

分析： 该方程中未知数是 x，其他变量视为常数。程序如下：

```
x = sym('x');                  % 定义符号变量
f = sym('a*x^2+b*x+c=0');        % 求方程的解
% f = 'a*x^2+b*x+c=0';           % 写为字符串形式也可
x = solve(f)                   % 利用 solve 求方程的根
pretty(x)                      %  把所求结果转化为手写形式
xL = latex(x)                  % 把结果转为 Latex 命令，复制这些内容到 CTeX 软件可显示为公式
```

程序输出下面的结果：

```
x = [ 1/2/a*(-b+(b^2-4*a*c)^(1/2))]
[ 1/2/a*(-b-(b^2-4*a*c)^(1/2))]
                    [           2         1/2]
                    [    -b + (b  - 4 a c)    ]
                    [1/2 --------------------]
                    [            a           ]
                    [                        ]
                    [           2         1/2]
                    [    -b - (b  - 4 a c)    ]
                    [1/2 --------------------]
                    [            a           ]
xL =
\left [\begin {array}{c} 1/2\,{\frac {-b+\sqrt
{{b}^{2}-4\,ac}}{a}}\\\noalign{\medskip}1/2\,
{\frac{-b-\sqrt {{b}^{2}-4\,ac}}{a}}\end {array}\right ]
```

这个结果与用如下二次方程求根公式得到的结果是一致的。

$$x_{1,2} = \frac{b \pm \sqrt{b^2 - 4ac}}{2a}$$



这里用一个错误的语句 x = solve('ax^2+bx+c=0')将会返回 x = -ax^2-bx，MATLAB 把“ax”、“bx”和“c”认为是变量，因为按字母顺序“c”排在最后，“c”将被认为是未知数，而把“ax”和“bx”作为常数。因此读者在使用 solve 函数时，方程表达式一定要正确地输入，否则会引起不必要的麻烦。

例 8-2：分别求解方程 $e^{-px}+b\sin a=0$ 和 $3a+4\cos(a+\pi)=0$，其中 a 为未知数。

分析：第一个方程中将 p，x 和 b 作为常数看待，第二个方程中含有 π（它可以转换为数值）。程序如下：

```
E1='exp(-p*x)+b*sin(a)';          % 定义方程
E2='3*a+4*cos(a+pi)=0';          % 定义方程
a1=solve(E1,'a')                 % 求解方程，并指定 a 为未知数
a2=solve(E2,'a')                 % 求解方程，并指定 a 为未知数
a2d = double(a2)                 % 转化为双精度数据
```

求解得到：

```
a1 =-asin(exp(-p*x)/b)
a2 =.86492728054203081492303563791032
a2d =0.8649
```

例 8-3：求下面的非线性 4 元方程组的解。

$$\begin{cases} a+b+x=3y \\ ax-by=1 \\ ab+xy=2 \\ a+b=(x+y)^2 \end{cases}$$

分析：第 1 个方程是线性的，而其他 3 个是非线性方程。计算程序如下：

```
E1 = 'a+b+x=y*3';              % 定义方程
E2 = 'a*x-b*y=1';              % 定义方程
E3 = 'a*b+x*y=2';              % 定义方程
E4 = 'a+b=(x+y)^2';             % 定义方程
[a, b, x, y] = solve(E1,E2,E3,E4);   % 解方程组
an = numeric(a);                % 把 a 转化为数据型
bn = numeric(b);                % 把 b 转化为数据型
xn = numeric(x);                % 把 x 转化为数据型
yn = numeric(y);                % 把 y 转化为数据型
Solution = [an, bn, xn, yn]         % 合为一个矩阵
```

计算结果如下:

```
Solution =
  -0.6669 + 1.0755i  -1.0980 - 1.8069i   0.6435 - 0.8339i  -0.3738 - 0.5217i
  -0.6669 - 1.0755i  -1.0980 + 1.8069i   0.6435 + 0.8339i  -0.3738 + 0.5217i
  -0.2447 + 1.4823i  -0.2765 - 1.3702i   0.0724 - 0.5726i  -0.1496 - 0.1535i
  -0.2447 - 1.4823i  -0.2765 + 1.3702i   0.0724 + 0.5726i  -0.1496 + 0.1535i
   0.2183 + 0.2785i   0.7989 - 3.3011i  -1.3420 + 1.5372i  -0.1083 - 0.4951i
   0.2183 - 0.2785i   0.7989 + 3.3011i  -1.3420 - 1.5372i  -0.1083 + 0.4951i
   2.9578             0.4401             0.5330             1.3103
   6.9289             0.2111             0.2191             2.4530
```



矩阵 Solution 各列依次表示未知数 a，b，x 和 y。Solution 中有 8 行，表明方程组有 8 组解，其中前 6 组解都是复数，最后两行是实数解。此外颠倒输出变量的次序不影响结果，即 [a, b, x, y] = solve(E1,E2,E3,E4) 与 [y, x, b, a] = solve(E1,E2,E3,E4)得到的解是一样的。

例 8-4：求解下面的非线性方程组。

$$\begin{cases} xy = 1 \\ x^y = 4 \end{cases}$$

程序如下:

```
syms x y;      % 定义符号变量
E1 = 'x*y=1';    % 定义方程
E2 = 'x^y=4';    % 定义方程
J = solve(E1,E2,x,y);
% 求解方程，J 是结构体
x=J.x              % 显示 x
y=J.y              % 显示 y
```

上述程序计算得到下面的结果:

```
x = -1/log(4)*lambertw(-log(4))
y =-log(4)/lambertw(-log(4))
```

结果中的lambertw是一个特殊函数，MATLAB 中 w=lambertw(X)得到的是方程 $we^w = X$ 的解。图 8.1 给出的是这个函数的解。在 $x > -0.3678$ 时，此函数返回的是实数。对本问题的方程，使用 double 函数或者 numeric 函数可以得到数值解 x=0.0640-1.0908i 和 y=0.0536+0.9136i。

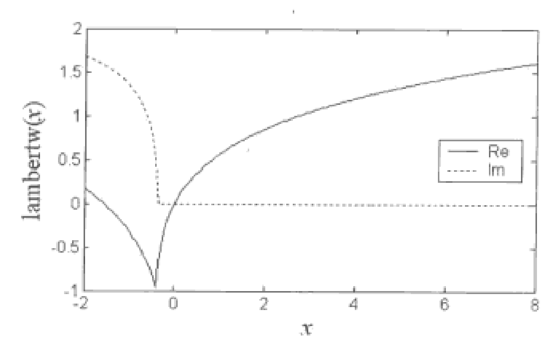


图 8.1 lambertw 函数曲线

例 8-5：求方程 $x^3+bx^2+cx+d=0$ 的解。

分析：一般的 3 次方程 $ax^3+bx^2+cx+d=0$ 可以转化为 $x^3+bx^2+cx+d=0$，这里我们利用函数 solve 来得出相应的求根公式。程序如下：

```
syms b c d x;                          % 定义变量
t = solve(x^3+b*x^2+c*x+d=0);          % 解方程
[r,s] = subexpr(t,'s')                 % 简化方程
```

输出下面的结果：

```
r =
[                                      1/6*s(1)^(1/3)-6*s(2)-1/3*b]
[ -1/12*s(1)^(1/3)+3*s(2)-1/3*b+1/2*i*3^(1/2)*(1/6*s(1)^(1/3)+6*s(2))]
[ -1/12*s(1)^(1/3)+3*s(2)-1/3*b-1/2*i*3^(1/2)*(1/6*s(1)^(1/3)+6*s(2))]
s =
[        36*c*b-108*d-8*b^3+12*(12*c^3-3*c^2*b^2-54*c*b*d+81*d^2+12*d*b^3)^(1/2
)]
[(1/3*c-1/9*b^2)/(36*c*b-108*d-8*b^3+12*(12*c^3-3*c^2*b^2-54*c*b*d+81*d^2+12*d
*b^3)^(1/2))^(1/3)]
```



这里使用函数subexpr进行变量替换来缩短表达式，实际上t是一个很长的表达式。

在使用函数 solve 的时候，方程表达式中等于 0 的部分可以不写，如求解下面的方程：

$$\begin{cases}x^2+y^2=8\\x+y=0\end{cases}，等价于\begin{cases}x^2+y^2-8=0\\x+y=0\end{cases}$$

程序如下：

```
syms x y;                                    % 定义符号变量
[x,y] = solve('y^2+x^2-8', 'y+x', x, y);     % 仅写求解方程组中的 f(x,y) 部分
solutions = [x,y]                            % 把解合并为一个矩阵，每一个表示一组解
```

得到下面的解：

```
solutions =
[ -2,  2]
[  2, -2]
```

可见“=0”在编程时可以不写，函数 solve 默认这部分存在。然而不是所有方程都能用 solve 得到结果的，下面是几个例子。

输入命令：

```
>> x = double(solve('x*sin(x)'))
```

输出结果是：

```
x =     0
```



由于知道方程 $x\sin x = 0$ 的通解是 $x = n\pi$，其中 n 是整数，即 solve 函数可能发生“漏解”现象。

输入命令：

```
>> solve('x^2-a=0')
```

输出结果：

```
ans =[  a^(1/2)]
[ -a^(1/2)]
```

输入命令：

```
>> solve('x^2-a*sin(x)+b=0')
```

输出结果：

```
Warning: Explicit solution could not be found.
> In C:\MathTools\MATLAB65\toolbox\symbolic\solve.m at line 136
ans =[ empty sym ]
```

输入命令：

```
>> solve('x^2-a*sin(x)+b*x=0')
```

输出结果：

```
??? Error using ==> solve
Error, (in allvalues/rootseq) cannot evaluate with symbolic coefficients
```

可见表达式复杂的时候，solve 函数可能会给不出相应的结果。

8.1.2 求解非线性方程

MATLAB 提供的 fzero 函数是计算非线性方程的根，这个函数将返回给定初值附近的根。当多个根的时候，fzero 函数需要多次指定初值。该函数调用格式为：

```
x = fzero(fun, x0)
x = fzero(fun, x0, options)
[x, fval, exitflag, output] = fzero(fun, x0, options)
```

参数说明：x 是返回的根。fval 是 x 处目标函数的值。exitflag 表明解存在的情况，即正数表明

解存在，负数表示解不存在（遇到复数、NaN 或者无穷大等）。参数 output 包含计算过程中的信息，它是一个结构体，output.algorithm 是所用的算法，output.funcCount 是函数赋值次数，output.iterations 是迭代次数。fun 表示函数表达式。x0 是初始值，它只能是一个有限的实数。options 是设置过程参数，它可以通过函数 optimset 来设定参数，即 options = optimset('fzero')。函数 fzero 在搜索区间内发现 inf、nan 或复数值时中止搜索。如果搜索失败，则返回 Nan，并显示：

```
Exiting fzero: aborting search for an interval containing a sign change
    because NaN or Inf function value encountered during search
(Function value at -1.716199e+154 is Inf)
Check function or try again with a different starting value.
```

因此，可以知道使用 fzero 函数是不能得到方程的复数解的，而 solve 函数可以得到复数解。

如果初值 x0 是一个长度为 2 的矢量，则 fzero 函数假设 x0 是一个区间，其中 fun(x0(1))的符号与 fun(x0(2))的不同。如果符号相同，则会显示出错。给出符号不同的区间可以保证 fzero 函数返回一个 fun 函数改变符号的位置附近的值。函数 fun 的定义在 7.0 版本以后支持匿名函数，即下式是合法的。

```
x = fzero(@(x)x.^3-2*x+6,1);
x = fzero(@humps,1.2)
```

其中函数 humps 是 MATLAB 自带函数，表达式为：

$$\mathrm{humps}(x)=\frac{1}{(x-3)^2+0.01}+\frac{1}{(x-9)^2+0.04}-6$$

例 8-6： 求关于 x 的函数的零点，其中 $a=1$，$b=-2$。

分析： 上一小节中已经说明 solve 不能求解这个函数，这里用函数 fzero 来求解这个方程。程序如下：

```
syms x a b;                  % 定义符号变量
fx = x.^2-a*sin(x)+b;          % 定义符号函数
fx = subs(fx,{a,b},{1,-2});   % 对 a 和 b 赋值
fx = char(fx);                % 把 fx 转换为字符串
fx = strrep(fx,'^','.^');         % 把乘方运算转换为点运算
fx = inline(fx);               % 定义内联函数
x1 = fzero(fx,0)              % 求函数第 1 的零点
x2 = fzero(fx,2)              % 求函数第 2 的零点
xt = -6:.02:6;               % 产生[-6, 6]区间内等间距采样点
plot(xt,fx(xt));               % 绘制 fx 的曲线图
hold on;
plot(xlim,[0,0],'k:')           % 绘制 0 刻度线
```

上述程序输出：

```
x1 =   -1.0615
x2 =    1.7285
```

可见调用函数 fzero 计算，每次只能返回一个零点（根或者解）位置。为了得到所有的根，需要画出可能范围内 $f(x)$ 的曲线图（如图 8.2 所示）。从图中查看曲线与零线的交点数目和大概位置来确定初值的选择，否则会漏掉一些根。

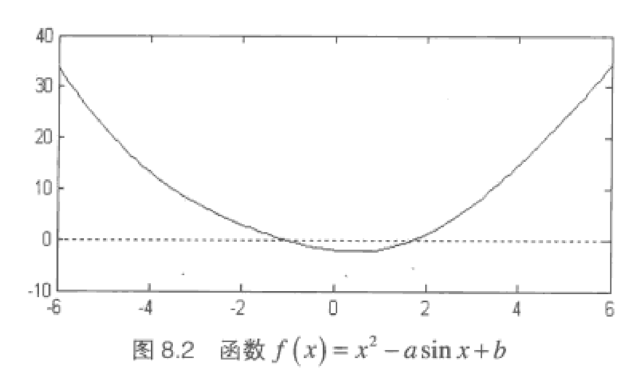


图 8.2 函数 $f(x)=x^2-a\sin x+b$

例 8-7：不同初值对求解方程 $f(x)=\sin x^2=0$ 的影响，其中初值选择 x0 = -0.1, 0, 0.1，如图 8.3 所示。

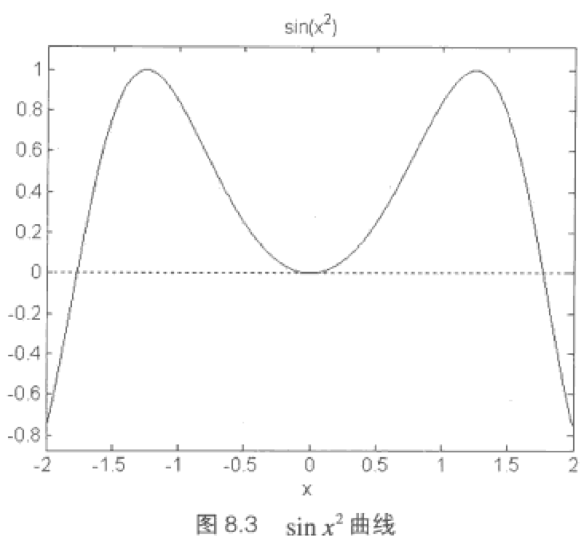


图 8.3 $\sin x^2$ 曲线

分析：从这个例子中可以发现返回的结果并不是与初值最近的点。程序如下：

```
x01 = fzero('sin(x^2)',0)       % 初值为 0
x02 = fzero('sin(x^2)',0.1)      % 初值为 0.1
x03 = fzero('sin(x^2)',-0.1)     % 初值为-0.1
set(gca,'Fontsize',12,'Fontname','Times new roman');
ezplot('sin(x^2)',[-2,2]);        % 绘制 sin(x^2)曲线
hold on;
plot(xlim,[0,0],'k:');            % 画 0 刻度线
```

输出如下：

```
x01 =      0
x02 =   -1.7725
x03 =    1.7725
```

可见初值的选择有时会影响函数的结果。另外从例 8-6 和例 8-7 的求解过程中，可以发现利用函数 fzero 求解结合目标函数绘图的方式是解方程较好的方式。

8.1.3 求解多元非线性方程

在 MATLAB 的最优化工具箱里提供了函数 fsolve 求解多元非线性方程，其调用格式为：

```
x = fsolve(fun, x0);
x = fsolve(fun, x0, options);
x = fsolve(fun, x0, options, p1, p2,...);
[x, fval] = fsolve(fun, x0);
[x, fval, existflag] = fsolve(fun, x0);
[x, fval, existflag, output] = fsolve(fun, x0);
[x, fval, existflag, output, Jacob] = fsolve(fun, x0);
```

参数说明： x 返回方程组的解。fval 是目标函数在 x 处的值。existflag 反应解的情况，意义同函数 fzero。output 返回输出信息，具体包含迭代数 output.iterations、函数计算次数 output.funcCount、使用算法 output.algorithm、CG 迭代数 output.cgiterations 以及第一阶优化 output.firstorderopt。Jacob 是函数在 x 处的雅可比矩阵。fun 为定义好的非线性方程（组）的文件名（或函数名）。x0 为求解方程的初始向量。options 设置求解过程的各种参数，一般采用默认参数 options = optimset('fsolve')。参数 p1，p2 等是函数 fun(x, p1, p2,⋯)中的参数。

例 8-8： 求解 MATLAB 自带的啁啾函数 chirp 的零点，其中参数 F0 = 0，T1 = 5，F1 = 1。变量 $x\in[0,5]$，要求得出这个区间内所有零点。

分析： 啁啾信号是频率逐渐变大的一类时间信号，在信号处理中经常使用。这个函数多次穿过 0 刻度线。这里使用初值 1，2.5，3.5，4，4.7。程序如下：

```
F0 = 0;          % 定义 t=0 处的频率 F0
T1 = 5;          % 定义时间 T1
F1 = 1;          % t=T1 处的频率 F1
options = optimset('Display','iter');                          % 设置参数选项
t = fsolve(@chirp,[1,2.5,4,3.5, 4.7], options, F0,T1,F1)   % 求解方程
chirp(t,T0,F1,T1)=0
tt = linspace(0,5,400);                                % 等间距采样
yy = chirp(tt,F0, T1, F1);                             % 计算各点函数值
figure;
plot(tt,yy,'r');                                       % 绘制啁啾曲线
hold on;
plot(xlim,[0,0],'k:');                                 % 画 0 刻度线
plot(t,zeros(size(t)),'b+');                           % 用符号 "+" 显示求解结果
set(gca,'Position',[0.13 0.51 0.775 0.315]);           % 重置坐标轴（axes）位置
```

上述程序计算得到下面的信息：

```
                        Norm of      First-order   Trust-region
 Iteration    Func-count      f(x)           step          optimality      radius
     1          6          1.8984                       2.39          1
     2         12        0.648322          1            1.94           1
     3         18       0.0076798       0.393723         0.164           1
     4         24     2.08742e-006      0.0445106       0.00287          1
     5         30     1.09496e-013     0.000726799     6.57e-007          1
Optimization terminated successfully:
 First-order optimality is less than options.TolFun.
t =    1.5811    2.7386    4.1833    3.5355    4.7434
Maximum number of function evaluations reached:
```

```
increase options.MaxFunEvals
```



从啁啾函数的曲线可以看出，在区间$[0,5]$内存在 5 个零点。利用预先设定的初始值，完全求出了其中的零点。这里进一步考察不同初始值收敛的结果（续上面的程序），如图 8.4 所示。

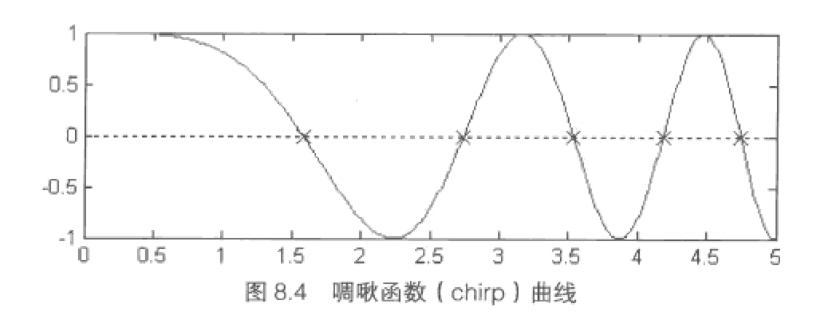


图 8.4 啁啾函数（chirp）曲线

```
options = optimset('fsolve');                    % 设置控制参数
tr1 = fsolve(@chirp,tt,options,F0,T1,F1);          % 利用向量方式求根计算
for k = 1:length(tt);
   tr2(k) = fsolve(@chirp,tt(k),options,F0,T1,F1);  % 利用循环方式求根计算
end
figure;
subplot(121);
plot(tt,tr1,'.');                                % 画出求解结果
subplot(122);
plot(tt,tr2,'.');                                % 画出求解结果
```

此处略去部分关于图形设置程序，详细的内容可参阅光盘中的文件 example_8.m，输出结果如图 8.5 所示。

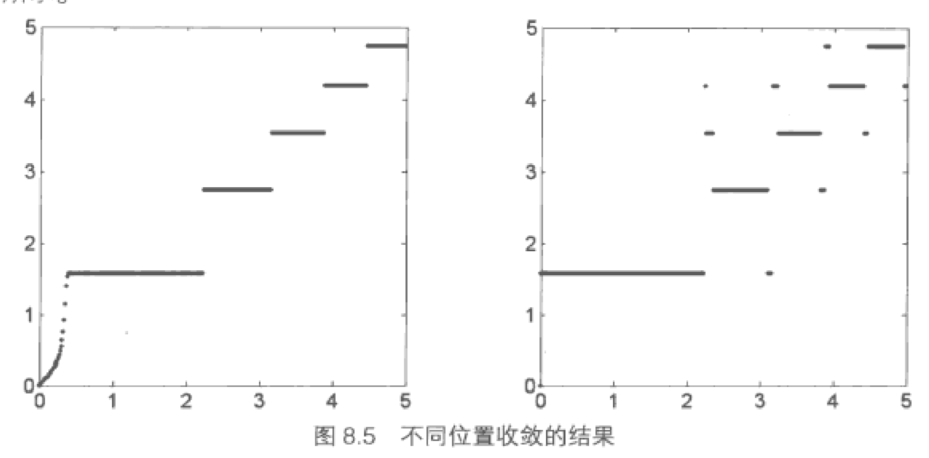


图 8.5 不同位置收敛的结果

两幅图形大体相同，但是在 0 附近两幅图存在着明显差异。此外在右图中存在着离散的小线段。这是因为采用向量方式计算每次迭代次数受限制，有的点没有完全达到收敛而停止。因此，在计算中应该区别使用这两种方式，此外全域绘图还是很重要的，它可以检查计算中的疏漏。

例 8-9：利用 fsolve 函数求解下面的二元非线性方程。

$$\begin{cases} x^2\cos(2x)+y^2\sin(2y)=1 \\ x^3+y^3-6\cos(2xy)=-1 \end{cases}$$

其中 $x, y \in [-2,2]$。要求得出定义域内所有的解。

分析：其中的两个方程是较复杂的函数。在求解之前我们很难知道解的个数和大致位置，可以调用函数 ezplot 绘制 $f_1(x,y)=x^2\cos(2x)+y^2\sin(2y)-1$ 和 $f_2(x,y)=x^3+y^3-6\cos(2xy)+1$ 对应的曲线，即：

```
ezplot('y^2*sin(2*y) + x^2*cos(2*x) -1',[-2,2]);        % 绘制第一个方程曲线
h=get(gca,'Children');                                  % 获取句柄信息
set(h,'Color','r','LineStyle',':');                     % 改变曲线颜色和线型
hold on;
ezplot('x^3 + y^3 - 6*cos(2*x*y) + 1',[-2,2]);          % 绘制第一个方程曲线
str=get(get(gca,'Title'),'String');                     % 获取当前 title 中的字符串
title(['x^2cos(2x)+y^2sin(2y)-1=0, ',str]);             % 更换字符串
axis equal;                                             % 设置单位长度一致
```

根据上面函数绘制的曲线图形如图 8.6 所示。

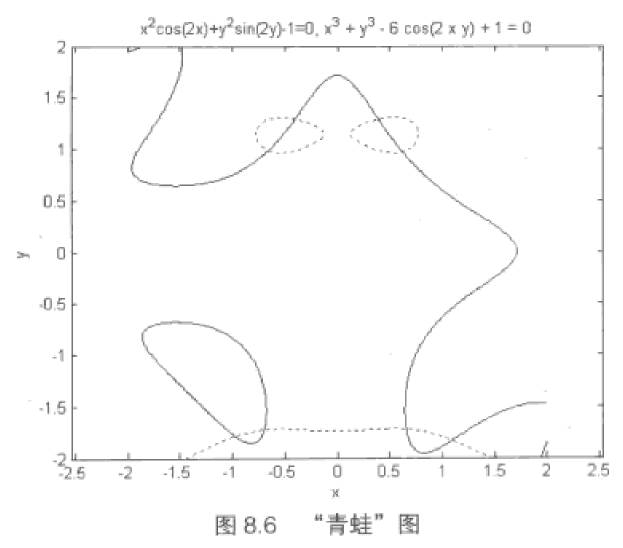


图 8.6 “青蛙”图

图 8.6 外形酷似一只“青蛙”，红色虚线表示第 1 个方程的曲线，而蓝色实线对应于第 2 个方程的曲线，其中的交点就是方程组的解。为了方便地获取两曲线交点处的坐标，可以使用函数 ginput 方便地使用鼠标得到交点附近的点作为初始值，即：

```
x0 = ginput(8)                              % 鼠标取点
```

在获取图 8.6 中 8 个点的大致位置之后，可以调用函数 fsolve 求解，程序如下：

```
Eq1 = 'x(1)^2*cos(2*x(1))+x(2)^2*sin(2*x(2))-1';  %第一个方程，x 对应于 x(1)，y 对应于 x(2)
Eq2 = 'x(1)^3+x(2)^3-6*cos(2*x(1)*x(2))+1';       %第二个方程，x 对应于 x(1)，y 对应于 x(2)
fun = ['[',Eq1,',',Eq2,']'];                        % 形成方程组
for k = 1:size(x0,1);                               % 逐点求根
```

```
    x = fsolve(fun,x0(k,:));                        % 用 fsolve 函数求根
    xr(k,:) = x;                                % 存储根的数值
end
xr                                          % 显示数据
plot(xr(:,1),xr(:,2),'k*','markersize',10);               % 把解画到图上
```

为了比较，我们把 x0 和 xr 的值显示于下：

```
x0 =                            xr =
   -0.3918    1.2924               -0.4007    1.2885
   -0.6842    0.9883               -0.6571    0.9817
   -0.6725   -1.7368               -0.6998   -1.7270
   -0.9415   -1.8070               -0.9744   -1.7887
    0.6140   -1.7135                0.6569   -1.7231
    1.1053   -1.8772                1.0939   -1.8343
    0.6140    0.9766                0.6219    0.9685
    0.4035    1.2456                0.3926    1.2866
```

x0 的数据可能会因为使用者点击位置的差异而有所不同，它可以认为是图解法的一个粗糙结果。而 xr 是利用函数 fsolve 得到的精确结果，它不因使用者点击位置不同而改变。为了进一步在视觉上证明所得结果的准确性，这里把 xr 的数据画到两个函数的曲线图上，如图 8.7 所示，可以发现各个点完好地吻合于两曲线的交点处。

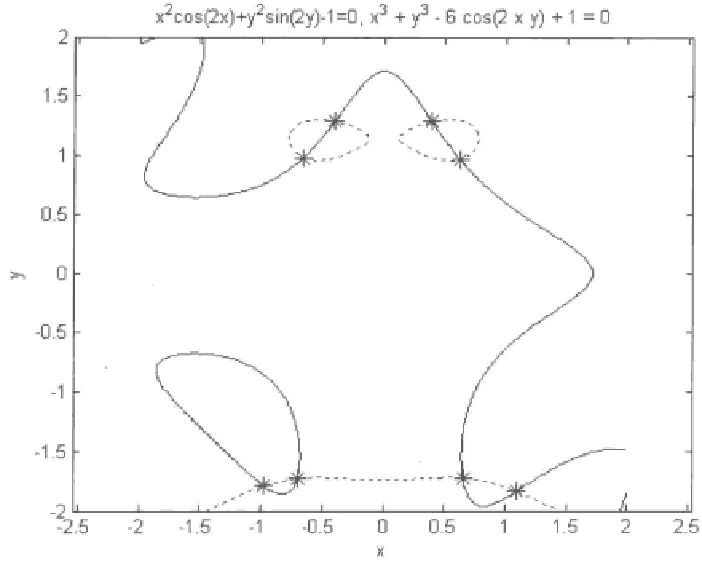


图 8.7　fsolve 函数求解之后的结果显示

8.1.4　求解多项式的根

MATLAB 提供函数 roots 来求解多项式的根，其调用格式如下：

```
x = roots(C);
```

参数说明： x 表示返回多项式函数的根。C 是多项式系数组成的向量，其中 C 的系数与多项式之间的关系是 C(1)*X^N + … + C(N)*X + C(N+1)，即 C 的系数对应于按降幂顺序排列的多项式前面的系数。如果多项式里面缺少某幂次项时，那么它前面的系数等于 0，如多项式

$P_4(x)=x^4+2x^2+4x+5$ 中，x^3 是缺少的，则 C 的值为[1, 0, 2, 4, 5]。此外 C 不能是矩阵。

例 8-10：求下面多项式的根。

$$P_3(x)=4x^3+3x^2+2x+1$$

$$P_5(x)=x^5+2x^2+x+10$$

$$P_7(x)=x^7+4$$

分析：根据三组多项式前面的系数，我们可以确定出系数向量 C，然后调用函数 roots 即可得到各个多项式的根。

相应计算程序如下：

```
C3 = [4,3,2,1];                % 定义系数向量
C5 = [1,0,0,2,1,10];           % 定义系数向量
C7 = [1,zeros(1,5),4];         % 定义系数向量
R3 = roots(C3)                 % 求多项式的根
R5 = roots(C5)                 % 求多项式的根
R7 = roots(C7)                 % 求多项式的根
```

结果如下：

```
R3 =                       R5 =                          R7 =
  -0.6058                    1.2545 + 1.1279i             -1.2190
  -0.0721 + 0.6383i          1.2545 - 1.1279i             -0.7600 + 0.9531i
  -0.0721 - 0.6383i         -1.6967                       -0.7600 - 0.9531i
                            -0.4062 + 1.3806i              0.2713 + 1.1885i
                            -0.4062 - 1.3806i              0.2713 - 1.1885i
```



对于多项式 $P_7(x)=x^7+4$ 的求根运算，我们可以推广至求解 n 次方根的运算，即 $\sqrt[n]{x_0}$ 可以转化为求解多项式 $P_n(x)=x^n-x_0$ 的根，可以使用语句 roots([1, zeros(1, n-1), x0]) 来计算。

与函数 roots 相对的函数是 poly，它是根据多项式的根反过来确定多项式的，其调用格式为：

```
x1 = poly(V);
x2 = poly(A);
```

参数说明：其中 V 是向量。A 是矩阵。x1 是构造的多项式系数。x2 是矩阵 A 的特征多项式（$\det(\lambda E_n-A)$，其中 E_n 是 n 阶单位矩阵、A 是一个 $n\times n$ 的方阵）的系数，其长度是 $n+1$。

比如通过例 8-10 中得到的 R5，利用函数 poly 来恢复系数。

```
cr5 = poly(R5)
cr5 =1.0000    0.0000   -0.0000    2.0000    1.0000   10.0000
```

这个系数与 C5 是一致的，因此可以认为函数 roots 和函数 poly 互为反函数。

对于由多项式组成的分式，MATLAB 提供函数 residue 实现部分分式展开，该函数调用格式为：

```
[R, P, K] = residue (B, A); % 格式 1
[B,A] = residue (R, P, K); % 格式 2
```

参数说明： 其中格式 1 和格式 2 是互为相反展开的操作。参数 R，P，K，B 和 A 的意义如 $\frac{B(s)}{A(s)}=\frac{\mathrm{R}(1)}{s-\mathrm{P}(1)}+\frac{\mathrm{R}(2)}{s-\mathrm{P}(2)}+\cdots+\frac{\mathrm{R}(n)}{s-\mathrm{P}(n)}+\mathrm{K}(s)$，其中参数 B，A 和 K 分别是多项式 $B(s)$，$A(s)$ 和 $K(s)$ 的系数，n 是多项式 $A(s)$ 的最高次数。特别地，如果满足 P(j)=⋯=P(j+m-1)，相应的分式项变为 $\frac{\mathrm{R}(\mathrm{j})}{s-P(\mathrm{j})}+\frac{\mathrm{R}(\mathrm{j}+1)}{[s-\mathrm{P}(\mathrm{j})]^2}+\cdots+\frac{\mathrm{R}(\mathrm{j}+\mathrm{m}-1)}{[s-\mathrm{P}(\mathrm{j})]^{\mathrm{m}}}$。

8.1.5 fminbnd 函数

接下来介绍 fminbnd 函数，在使用这个函数之前，我们需要将解方程问题转化为求最小值问题。如果函数 $f(x)$ 在区间 $[x_1,x_2]$ 存在零点，那么函数 $f'(x)=|f(x)|$ 的最小值一定是 0，且对应的位置就是 $f(x)=0$ 的解。函数 fminbnd 的调用格式为：

```
x = fminbnd(fun,x1,x2);
```

参数说明： x 是返回的最小值。fun 是函数表达式。x1 和 x2 表示函数定义区间[x1, x2]。

例 8-11： 求下面函数在 $[1,2]$ 区间内的零点。

$$f(x)=\frac{x^2-4\sin 5x}{x^2+\cos x}$$

分析： 首先把问题转化为求最小值问题，然后调用函数 fminbnd 求解。程序如下：

```
fun = inline('abs([x^2-4*sin(5*x)]/[x^2+cos(x)])');
                              % 转化为函数 f(x)的绝对值
x01 = fminbnd(fun,1,1.5)         % 计算在区间[1, 1.5]内最小值
x02 = fminbnd(fun,1.5,2)         % 计算在区间[1.5, 2]内最小值
plot(x01,0,'r*','markersize',14);    % 把解画到图上
plot(x02,0,'r*','markersize',14); % 把解画到图上
```

略去绘图程序，完整程序见光盘中的 example_11.m 文件。为了比较结果的准确性，将所得结果画为如图 8.8 所示，可见函数 fminbnd 得到了较精确的结果。

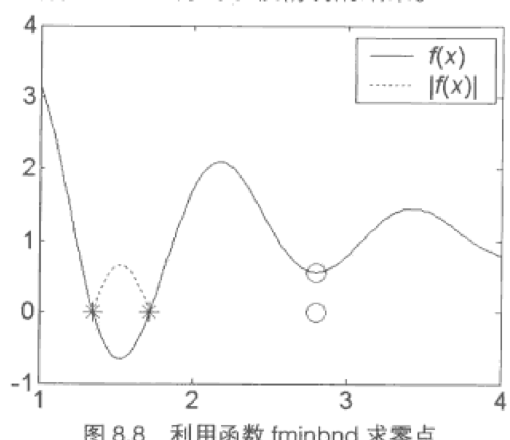


图 8.8 利用函数 fminbnd 求零点



在使用 fminbnd 函数的时候要求函数曲线在定义范围内的极小值必须小于 0，否则会引起麻烦。如语句 x03 = fminbnd(fun,1,4)，得到 x03 =2.8009。

很明显 x03 这一点是函数 $f(x)$ 在区间[1, 4]内的第二个最小值位置，即函数 fminbnd 搜索到的最小值是第二个极小值。如果在调用函数 fminbnd 之前使用 plot 函数先绘制曲线，则根据曲线上零点的大致位置确定区间的端点值，可以避免出现错解的现象。

8.2 数值方法

前面介绍了利用 MATLAB 自带函数来计算超越方程的根。每个函数都存在一定局限性，本节介绍几种数值方法，包括二分法、抛物线法、牛顿法、正割法和 Steffenson 法来求解超越方程。

8.2.1 二分法

在所有数值方法中，二分法是超越方程求解最直观、最简单的方法。二分法是以连续函数的介值定理为基础建立的。由介值定理可知，如果函数 $f(x)$ 在区间 $[a,b]$ 上连续，同时 $f(a)f(b)<0$，即 $f(a)$ 和 $f(b)$ 符号相反，那么 $f(x)$ 在区间 $[a,b]$ 内一定有实根。

二分法的基本思想是：用对分区间的思路根据分点处函数 $f(x)$ 的符号逐步将有限区间缩小，使在足够小的区间内，分点与理论上的根之间距离足够小，继而用分点的值近似理论上的根。

图 8.9 给出了二分法的主要流程。首先判断在区间端点 a 和 b 处的函数值是否异号，如果符号相同则直接输出“方程无解”。如果两端点函数值异号，那么将进入二分法处理程序：得到中点坐标 m（$m = [a+b]/2$）；判断中点函数值 $f(m)$ 与端点函数值 $f(a)$ 和 $f(b)$ 的关系，如果 $f(a)$ 和 $f(m)$ 同号，那么 a 的值由 m 更新，否则 b 的值由 m 更新；判断当前端点之间的距离是否小于预置精度 D（其值是一个很小的正数），如果 b-a>D，程序将返回到第一步继续二分过程，否则将输出 m 作为方程的近似解。该方法中不断二等分区间，因此称之为二分法。

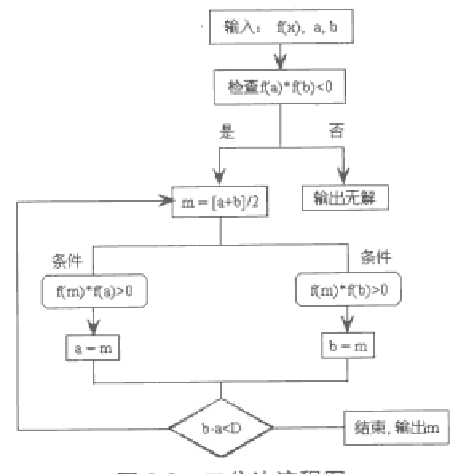


图 8.9　二分法流程图

假设等分 n 次，那么二分法的误差可以用 $\Delta < b-a$ 估计，其中 a 和 b 是程序结束时的数值，而非初始时的数值。

根据图 8.9 描述的流程，编写了二分法程序 dichotomy.m，该程序是一个函数文件，读者可以利用它直接求解一元方程。该程序调用格式为：

```
x0 = dichotomy('fun', a, b, D);
x0 = dichotomy('fun', a, b, D);
[x0, fval] = dichotomy('fund1', a, b, D);
```

参数说明：x0 是返回的方程的解。a 和 b 是区间的端点。D 是求解的精度，即一个很小的正数，其默认值是 1e-6。fun 表示这个程序中求解函数的定义函数文件名称，读者可以根据自己的需要把求解表达式 $f(x)=0$ 中的 $f(x)$ 输入到一个函数中。

例 8-12：求解下面两个方程在区间 $[1,4]$ 内的解。

$$x-2+\left(\int_0^{\sin 6x}\sqrt{t}e^{-t}\sin t\mathrm{d}t\right)^2=0，\quad y(t)=0$$

其中 $y''-(1-y^2)y'+y=0$，初始条件为 $y(0)=2$，$y'(0)=0$。

分析：在这个方程中包含一个变上限积分，这里将使用函数 quad 来离散积分。相关子函数如下定义：

```
function y=fund1(x);          % 用子函数定义f(x)
Ix=quad('sqrt(t).*sin(t).*exp(-t)',0,sin(6*x));
y=x-2+Ix^2;
```

第 2 个方程是 MATLAB 自带函数 vdp1 对应的微分方程，调用 ode45 函数即可对这个函数求解，具体函数定义如下：

```
function y = fund2(x);
[t,y] = ode45(@vdp1,[0, x], [2, 0]);
y = y(end,1);
```

然后调用函数 dichotomy 计算两个方程的解。执行下面的语句：

```
[x01, f1] = dichotomy('fund1',1,4)
[x02, f2] = dichotomy('fund2',1,4)
```

得到下面的结果：

```
x01 =    2.0089
f1 =  1.0263e-006
x02 =    2.1617
f2 =  6.0831e-007
```



可见使用二分法可以求解含有复杂表达式的方程，如果使用前面的 MATLAB 函数计算起来可能相当麻烦，甚至是不可能的。函数曲线与求解结果显示于图 8.10 中，可见所定义的二分法程序很好地解决了复杂方程的求解。

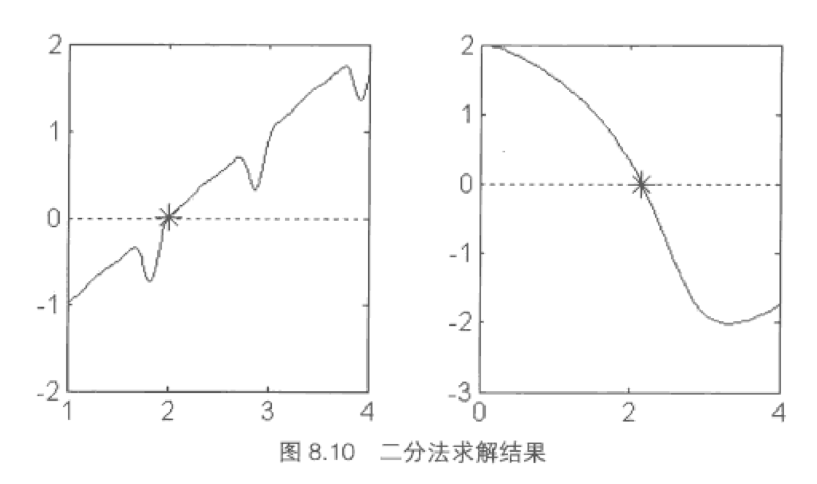


图 8.10　二分法求解结果

8.2.2　抛物线法

前面介绍了二分法利用两点来逼近准确的根，这里使用三点来逼近准确的根，即抛物线法。假设方程 $f(x)=0$ 的根为 x'。设迭代计算相邻的 3 个点的坐标是 x_{k-2}， x_{k-1} 和 x_k。下面将利用这三点推导获得下一点坐标 x_{k+1}。记 $f(x_{k-2})=f_{k-2}$， $f(x_{k-1})=f_{k-1}$ 和 $f(x_k)=f_k$，则过这三点的抛物线可以写为：

$$p_2(x)=f_k+\frac{f_k-f_{k-1}}{x_k-x_{k-1}}(x-x_k)+\frac{\dfrac{f_{k-2}-f_k}{x_{k-2}-x_k}-\dfrac{f_k-f_{k-1}}{x_k-x_{k-1}}}{x_{k-2}-x_{k-1}}(x-x_k)(x-x_{k-1}) \tag{8-1}$$

显然， $p(x_k)=f_k$， $p(x_{k-1})=f_{k-1}$。把 $x=x_{k-2}$ 带入上式，经简单计算可以得出 $p_2(x_{k-2})=f_{k-2}$，因此可以知道式(8-1)通过曲线上 3 个点 (x_{k-2},f_{k-2})，(x_{k-1},f_{k-1}) 和 (x_k,f_k)。这里略去复杂的推导过程。

在抛物线法中，使用多项式 $p_2(x)$ 来近似函数 $f(x)$。这里 $p_2(x)$ 是一个二次多项式，一般情况下它有两个根，我们选择离 x_k 较近的根作为下一个近似的根。考虑到使 $|x_{k+1}-x_k|$ 取到最小值，把式（8-1）写为另一种等价的形式：

$$p_2(x)=f_k+\frac{f_k-f_{k-1}}{x_k-x_{k-1}}(x-x_k)+\frac{\dfrac{f_{k-2}-f_k}{x_{k-2}-x_k}-\dfrac{f_k-f_{k-1}}{x_k-x_{k-1}}}{x_{k-2}-x_{k-1}}\left[(x-x_k)^2-(x_k-x_{k-1})(x-x_{k-1})\right] \tag{8-2}$$

令 $p_2(x)=0$，式（8-2）对应的方程可以写为下面的形式：

$$a_k+b_k(x-x_k)+c_k(x-x_k)^2=0 \tag{8-3}$$

其中 $a_k=f_k$， $c_k=\frac{\dfrac{f_{k-2}-f_k}{x_{k-2}-x_k}-\dfrac{f_k-f_{k-1}}{x_k-x_{k-1}}}{x_{k-2}-x_{k-1}}$， $b_k=\frac{f_k-f_{k-1}}{x_k-x_{k-1}}+c_k(x_k-x_{k-1})$。

在一元二次方程（8-3）中，当 $f_k=a_k=0$ 时，可得 $x=x_k$，即此时 x_k 就是方程的解，可以

终止迭代。

当 $f_k = a_k \neq 0$ 时，$x \neq x_k$ 成立同时解下面的方程：

$$a_k\left(\frac{1}{x-x_k}\right)^2 + b_k\frac{1}{x-x_k} + c_k = 0$$

可得：

$$\frac{1}{x-x_k} = \frac{-b_k \pm \sqrt{b_k^2 - 4a_k c_k}}{2a_k} \tag{8-4}$$

这里不求关于 $(x-x_k)$ 的一元二次方程是因为无法确定 c_k 是否等于 0。式（8-4）可以进一步写为：

$$x - x_k = -\frac{2a_k}{b_k \mp \sqrt{b_k^2 - 4a_k c_k}}$$

或：

$$x = x_k - \frac{2a_k}{b_k \mp \sqrt{b_k^2 - 4a_k c_k}} \tag{8-5}$$

为了保证 x 和 x_k 之间的距离最近，需要使式（8-5）中分母最大，同时把 x 写为 x_{k+1}，即：

$$x = x_k - \frac{2a_k}{b_k + \operatorname{sgn}(b_k)\sqrt{b_k^2 - 4a_k c_k}} \tag{8-6}$$

这里使用符号函数保证分母中两项同号取绝对值最大，从而保证 x 和 x_k 之间的距离最近。式（8-6）又称为抛物线法的迭代公式。同时抛物线法又称为 Muller 法。在式（8-6）中，根号下面的表达式可能是负数，因此使用抛物线法得到的解可能是实数也可能是复数。在抛物线法中我们使用三点建立由一元二次函数获得的迭代公式，此外也可以使用更多点来建立迭代公式，但它们极少用到，还有两个原因：一是建立迭代函数需要用到高次多项式，其求根过程困难；二是收敛速度较慢，其收敛阶总是小于 2。

MATLAB 没有提供专门的函数实现抛物线法求解方程。根据上面的描述，作者编写了 parabola.m 程序实现抛物线法求解方程，其中迭代在满足条件 $|a_k| < D$（D 是预先指定的精度，默认值是 1e-6）时停止。另外还可以考虑使用条件 $|x_{k+1} - x_k| < D$ 控制程序执行，即相邻两点之间的距离足够小。函数 parabola 的调用格式为：

```
xr = parabola(fun,x0,x1,x2,D);
```

参数说明：其中 xr 是方程的近似解。fun 是待解函数对应的函数文件名。x0 是第一点的值，x1 是第二点的值，x2 是第三点的值，要求 x1<x2<x3。D 是预定义的精度。

例 8–13：利用抛物线法求解下面的方程在区间[1，3]内的解。

$$f(x) = x^3\cos x + 2x^2 - 2\sin x$$

分析：首先来定义这个函数。

```
function y = fund3(x);
y = x.^3.*cos(x)+2*x.^2-2*sin(x);
```

然后调用函数 parabola 就可以得到方程的解，相应程序如下：

```
xx = linspace(1,3,200);         % 对自变量采样
y = fund3(xx);                  % 计算个点的函数值
plot(xx,y);                     % 绘制函数 f(x)曲线
hold on;
plot(xlim,[0,0],'r:');          % 绘制 0 刻度线
xr = parabola('fund3',1,2,3)    % 抛物线法解方程
plot(xr,fund3(xr),'k*','markersize',12);  % 绘制解对应的点
```

上述程序输出：

```
xr =    2.3978
```

即该方程的解是 2.3978，同时可以把相应函数曲线和结果的数值显示在图 8.11 上，可见用抛物线法得到的结果是很好的近似解。

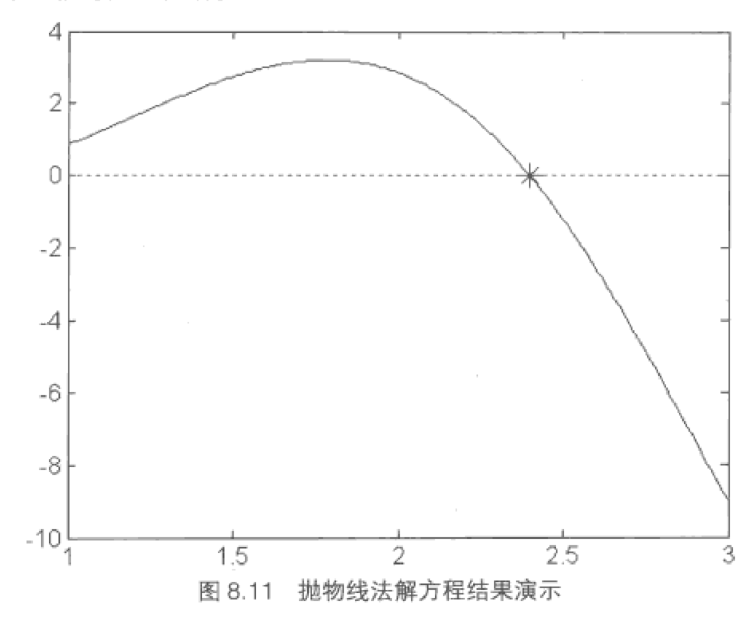


图 8.11 抛物线法解方程结果演示

8.2.3 牛顿法

前面介绍的是利用 2 个点或者 3 个点进行迭代得到的超越方程数值解法，这里考虑使用 1 个点进行迭代。根据函数 $f(x)$ 在 x_0 处的泰勒（Taylor）级数展开有：

$$f(x)=f(x_0)+f'(x_0)(x-x_0)+\frac{f''(\xi)}{2!}(x-x_0)^2,\quad \xi\in[x,x_0]$$

如果把 $(x-x_0)$ 看做高阶小量，那么：

$$f(x)=f(x_0)+f'(x_0)(x-x_0) \tag{8-7}$$

式（8-7）近似成立。求 $f(x)=0$ 可以等价地计算 $f(x_0)+f'(x_0)(x-x_0)=0$，进一步可以写为 $x=x_0-\dfrac{f(x_0)}{f'(x_0)}$。

上式可以写为下面的递推形式：

$$x_k=x_{k-1}-\frac{f(x_{k-1})}{f'(x_{k-1})} \tag{8-8}$$



在式（8-8）中可以看出要求 $f'(x)$ 在定义区间内不等于 0。

根据式（8-8），作者编写了用牛顿法求解超越方程的程序，函数文件名为 Newtonm。该函数的完整调用格式为：

```
xr = Newtonm(fun,x0,D);
```

参数说明： xr 是返回的解。fun 是定义的函数，可以用来计算 $f(x)$ 的函数值及其一阶导数值。x0 是初始值。D 是求解精度，默认值是 1e-6。

例 8–14： 用牛顿法计算下面方程在区间 $[3,5]$ 内的解。

$$f(x)=\frac{x^3\sin x}{10}+\frac{x^2}{4}-2\cos x$$

分析： 首先来定义函数 fund4 计算 $f(x)$ 的函数值及其一阶导数值。

```
function [y, dy, d2y] = fund4(x);
y = x.^3.*sin(x)/10+x.^2/4-2*cos(x);            % 计算函数值
if nargout>1;
   ff = sym('x^3*sin(x)/10+x^2/4-2*cos(x)');   % 定义符号函数
   dy = diff(ff);                          % 求一阶导数
   dy = subs(dy,x);                        % 赋值
end
if nargout==3;
   d2y = diff(ff,2);                        % 求二阶导数
   d2y = subs(d2y,x);                      % 赋值
end
```

然后调用函数 Newtonm 就可以求解这个超越方程了。程序如下：

```
xr1 = Newtonm('fund4',1.5)  % 牛顿法求根，初始点是 1.5
xr2 = Newtonm('fund4',3.5)  % 牛顿法求根，初始点是 3.5
xr3 = Newtonm('fund4',5.5)  % 牛顿法求根，初始点是 5.5
```

输出结果如下：

```
xr1 =    1.2679
xr2 =    4.0623
```

```
xr3 =5.9462
```



这里给出相应的函数曲线（如图 8.12 所示）。其中符号“□”、“*”“○”分别对应着初始值 1.5，3.5 和 5.5 的计算结果。从图 8.12 以及上面的结算结果可以发现，不同的初始点会引起结果的差异性。而且在一致单调的区间内才能得到期望的结果，即我们要求出 4.0623 这个解，需要在[3.3，5]区间内选择一点才有可能得到 4.0623 这个解。为了进一步明确这个规律，计算部分点作为初始值利用牛顿法来求解，同时相应的导数值也显示在表 8.1 中。

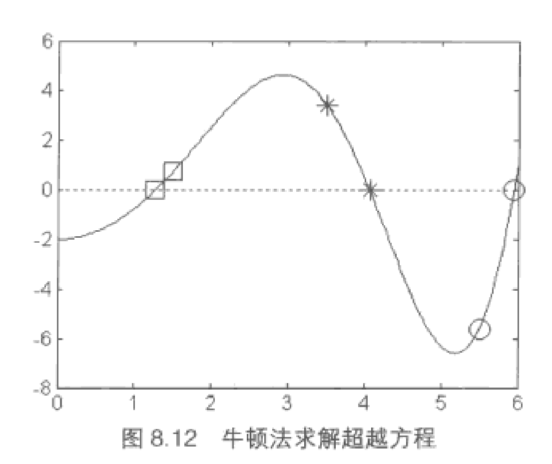


图 8.12 牛顿法求解超越方程

表 8.1 牛顿法解方程，初始值、解及一阶导数数据表

初始值 x_0	解	初始值处的一阶导数	初始值处的二阶导数	$f(x_0)f''(x_0)$
3.2	5.9462	-1.9673	-7.5509	32.9621
3.3	4.0623	-2.7296	-7.6726	-31.6922
3.4	4.0623	-3.4972	-7.6562	-29.2408
3.5	4.0623	-4.2558	-7.4885	-25.6964
3.6	4.0623	-4.9895	-7.1579	-21.2511
3.7	4.0623	-5.6816	-6.6550	-16.2043
3.8	4.0623	-6.3145	-5.9725	-10.9569
3.9	4.0623	-6.8700	-5.1063	-5.9979
4.0	4.0623	-7.3296	-4.0551	-1.8805
4.1	4.0623	-7.6749	-2.8206	0.8109
4.2	4.0623	-7.8878	-1.4085	1.5026
4.3	4.0623	-7.9509	0.1724	-0.3207
4.4	4.0623	-7.8481	1.9093	-5.0623
4.5	4.0623	-7.5644	3.7857	-12.9608
4.6	4.0623	-7.0870	5.7814	-24.0384
4.7	4.0623	-6.4050	7.8727	-38.0586
4.8	4.0623	-5.5102	10.0324	-54.4939
4.9	4.0623	-4.3972	12.2300	-72.5113
5.0	31.3367	-3.0640	14.4320	-90.9778
5.1	1.2679	-1.5118	16.6027	-108.4901

牛顿法收敛的充分条件为首先假设 $f(x)$ 在区间 $[a,b]$ 上有 2 阶连续导数，如果：

- $f(a)f(b)<0$，这一条件保证方程在该区间有根。
- 在整个区间 $[a,b]$ 上函数 $f(x)$ 的 2 阶导数不变号，同时 $f'(x)\neq 0$，该条件保证在区间内有唯一根。
- 选择的初始值 x_0 满足 $f(x_0)f''(x_0)>0$，保证产生的迭代序列单调有界且收敛。

表 8.1 中这个参数的取值是一种不满足充分条件的情况，可见牛顿法对于初始值 x_0 的选择比较苛刻，在使用这种方法的时候需要注意选择合适的初始值。

牛顿法的几何意义如图 8.13 所示（光盘中绘制该图程序的名称是 sketch_map1.m），线段 $\overline{x_1x_0}$ 的长度可以表示为：

$$\overline{x_1x_0}=\frac{f(x_0)}{\tan\alpha_1} \tag{8-9}$$

其中 $\begin{cases}\tan\alpha_1=f'(x_0)\\ \overline{x_1x_0}=x_0-x_1\end{cases}$，因此式（8-9）可以进一步写为 $x_0-x_1=\dfrac{f(x_0)}{f'(x_0)}$，即 $x_1=x_0-\dfrac{f(x_0)}{f'(x_0)}$。上式正好是牛顿法的迭代公式，按照这个过程递推下去就可以无限地逼近方程 $f(x)=0$ 的解。

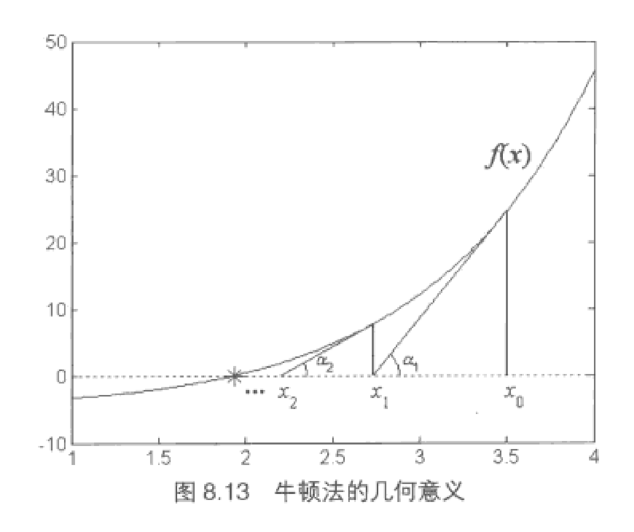


图 8.13 牛顿法的几何意义

8.2.4 正割法

前面介绍的牛顿法在每步计算中需要计算函数 $f(x)$ 及其一阶导数 $f'(x)$ 的数值，这相当于计算两个函数值，用时比较多。现在利用差商来代替牛顿法中的导数，如此一来可以减少一个函数值的计算。相应的牛顿法的迭代公式改为：

$$x_{k+1}=x_k-f(x_k)\frac{x_k-x_{k-1}}{f(x_k)-f(x_{k-1})} \tag{8-10}$$

正割法的几何意义如图 8.14 所示。在此图中，根据几何知识，三角形 $\Delta x_2 f_1 x_1$ 和三角形 $\Delta x_2 f_0 x_0$ 相似，则 $\dfrac{x_1-x_2}{x_0-x_1}=\dfrac{f_1}{f_0-f_1}$ 成立，可以进一步写为 $x_1-x_2=\dfrac{f_1\left(x_0-x_1\right)}{f_0-f_1}$，即 $x_2=x_1-\dfrac{f_1\left(x_0-x_1\right)}{f_0-f_1}=x_1-\dfrac{f_1\left(x_1-x_0\right)}{f_1-f_0}$。

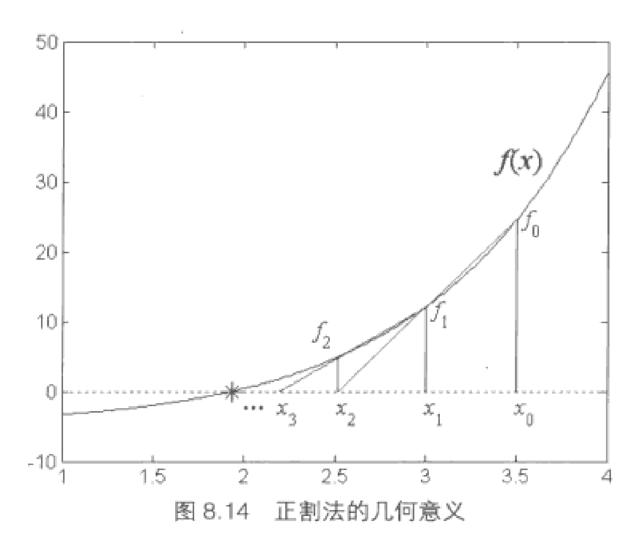


图 8.14　正割法的几何意义

上式正好是式（8-10）当 $k=1$ 时的表达式。如此递推下去就可以逐渐逼近方程 $f\left(x\right)=0$ 的解。这里同样地编写了函数 secant 来实现正割法求解方程的程序，其调用格式如下：

```
xr = secant(fun,x0,x1,D);
```

参数说明： xr 是返回方程的解。fun 表示方程中的函数 $f\left(x\right)$。x0 和 x1 是初始的两个值。D 是精度，默认值是 1e-6。

例 8-15： 利用正割法求下面方程的解。

$$f\left(x\right)=e^x-x-5=0$$

分析： 在求解的时候可以尝试不同类型初值的结果。程序如下：

```
fun = inline('exp(x)-x-5');          % 定义方程中的函数 f(x)
x0 = 3.5;                        % 第一初始值
x1 = 3;                          % 第二初始值
xr = secant(fun,x0,x1)                 % 用正割法解方程
xs = secant(fun,x1,x0)          % 用正割法解方程
xq = fzero(fun,2)              % 利用 fzero 函数比较
```

输出如下：

```
xr =    1.9368
xs =    1.9368
xq =    1.9368
```



在最初推导正割法的时候使用初始两点的关系是 $x_1 < x_0$，实际应用中这两点的大小关系不影响结果，此外解对 x_0 和 x_1 要求不高。正割法在有的教材中也称为割线法。

8.2.5　Steffenson 法

前面的正割法是使用差商代替导数，如果使用下面的形式代替导数：

$$\frac{f\left(x_k + f\left(x_k\right)\right) - f\left(x_k\right)}{f\left(x_k\right)}$$

可以得到 Steffenson 法的迭代公式：

$$x_{k+1} = x_k - \frac{\left[f\left(x_k\right)\right]^2}{f\left(x_k + f\left(x_k\right)\right) - f\left(x_k\right)} \tag{8-11}$$

用编写函数文件 steffenson.m 来实现 Steffenson 法解方程，其调用格式为：

```
xr = steffenson(fun, x1, D);
```

参数说明： xr 是返回方程的解。fun 表示方程中的函数 $f\left(x\right)$。x1 是初始值。D 是精度，其默认值是 1e-6。程序通过相邻的两个解 x_k 和 x_{k+1} 之间的距离控制迭代终止。

通过下面的语句计算例 8-15 中的方程：

```
xa = steffenson(fun,x1)
```

输出结果为：

```
xa =    1.9368
```



Steffenson 法解方程只要求一个初值，而且不需要计算导数值，但对初值选择比较敏感。我们测试了不同初值求解例 8-15 中的方程，相关数据如表 8.2 所示。

表 8.2　不同初值对 Steffenson 法解方程的影响

初始值	1.0	1.4	1.8	2.2	2.6	3.0	3.4
计算结果	-4.9932	5.5600	1.9368	1.9368	1.9368	1.9368	3.4000

8.3　小结

本章主要介绍了超越方程的解法。首先介绍了利用 MATLAB 提供的函数求解超越方程。函数 solve 是一种符号求解函数，利用这个函数我们可以得到一般方程的解，但是遇到较为复杂的方程时，这个函数就显得无能为力了。利用函数 fzero 和 fsolve 可以求解复杂函数在初值附近的解，在使用这两个函数的时候读者可以根据需要设定求解过程的参数。这两个函数只能求出方程的实数根。函数 roots 可用于计算多项式函数的根，一个特殊的应用就是求解某数的 *n* 次方根。如果把方程求根转化为求最小值问题，就可以调用函数 fminbnd 来求方程的根。接下来介绍了几种数值方

法求解超越方程。二分法在给定两个初值的条件下可以很快地得到方程的根，该方法对于初始条件要求低，即保证根存在就可以了。抛物线法需要给定 3 个初值再进行迭代计算。利用泰勒级数展开可以得到牛顿法求解方程的迭代公式。进一步地，利用差分替代导数可以得到正割法计算公式。类似地，给出 Steffenson 法计算公式。本章所述的无论哪一种方法，在利用的时候都需要结合画图才能完好地求解方程。因为现在计算机的运行速度很高，在进行数值计算时，计算量已经不影响方程的求解，不同方法彼此之间花费的时间差异很小，因此计算速度的问题不再讨论。

◆ ◆ ◆

第 9 章　数据拟合与插值

本章包括

- 拟合基础
- 多项式拟合
- Lagrange 插值
- 样条插值
- 最小二乘拟合
- 非线性拟合
- Hermite 插值
- 二维插值

在应用中，经常会遇到用一个解析函数描述已知数据（通常是测量值）的问题，而拟合与插值是解决这个问题的主要方法。拟合和插值在实验数据分析和处理中是十分重要的，特别是数据拟合，比如用最小二乘法求直线的斜率和截距是数据处理中经常使用的，而且很准确。此外外推法、标准工作曲线法都是这方面的运用。通过数据拟合，可以找到一条光滑的曲线来逼进数据。但由于运算量很大，长期以来使用手工绘图法来得到所需要的结果，其人为误差不可避免。如果是非线性的情况则更为复杂、困难，并很难得到所拟合变量间函数关系的解析式，也影响了对所研究问题的进一步深入。如果用 MATLAB 处理则非常方便，它可以画图和计算同时进行。除了 MATLAB 本身的拟合函数外，回归分析也可以用来解决数据拟合问题，这部分内容在第 11 章介绍。利用数据插值时，数据假定是准确的，要求以某种方法推测出数据点之间未知范围内的取值情况。插值法是一种用简单函数近似代替较复杂函数的方法，在插值点处的误差等于 0。

9.1　拟合基础

在很多科学研究中经常要根据实验数据建立数学模型。对于给定的数据需要用比较简单和满足相关物理意义的函数模型来逼近（或者称为拟合）实验数据。这种逼近的特点是：

- 需要适当的精度控制。
- 实验数据中由于一些人为和非人为因素而存在着小的误差；
- 对于一些问题，存在某些特殊信息能够帮助我们从实验数据中建立数学模型。

在介绍拟合方法之前，给出拟合结果优劣的评价方法。设 $\left[x_{1,k}, x_{2,k}, \ldots, x_{n,k}, y_k\right]$ 是测量数据，而函数关系是 $y=f\left(x_1, x_2, \ldots, x_n\right)$，那么拟合的数据值和实验数据值之间的误差（也可以称为残差）可以用下式计算。

- 绝对误差

$$\Delta_{\mathrm{a}}=\sum_k\left|f\left(x_{1,k}, x_{2,k}, \ldots, x_{n,k}\right)-y_k\right|$$

- 相对误差

$$\Delta_r = \frac{\sum_k \left| f\left(x_{1,k}, x_{2,k}, \ldots, x_{n,k}\right) - y_k \right|}{\sum_k \left| f\left(x_{1,k}, x_{2,k}, \ldots, x_{n,k}\right) \right|} \quad \text{或者} \quad \Delta_r = \frac{\sum_k \left| f\left(x_{1,k}, x_{2,k}, \ldots, x_{n,k}\right) - y_k \right|}{\sum_k \left| y_k \right|}$$

此外还可以通过均方误差 MSE（Mean square error）和信噪比 SNR（Signal noise ratio）来定义误差大小：

$$\mathrm{MSE} = \frac{\sum_k \left| f\left(x_{1,k}, x_{2,k}, \ldots, x_{n,k}\right) - y_k \right|^2}{\sum_k \left| f\left(x_{1,k}, x_{2,k}, \ldots, x_{n,k}\right) \right|^2} \text{或者} \mathrm{MSE} = \frac{\sum_k \left| f\left(x_{1,k}, x_{2,k}, \ldots, x_{n,k}\right) - y_k \right|^2}{\sum_k \left| y_k \right|^2}$$

$$\mathrm{SNR} = -20\log_{10}\mathrm{MSE}$$

上面介绍的参数都是单个值。为了得到整个范围内的误差情况，可以绘制图形来描述各点误差的情况。

在图 9.1 和 9.2 中使用了“+”来表示实际数据，而连续的曲线表示由拟合计算得到的数据。从这些图形可以很方便地观察出实验数据与相应的拟合值之间的差异大小。三维曲线由于存在着遮挡问题（如图 9.2(a) 所示），把曲线数据分别投影到 XY，YZ，ZX 平面上显示，这样就可以清楚地观察到。其中对于每一点的误差关系可以按 $\Delta = \sqrt{\Delta_x^2 + \Delta_y^2 + \Delta_z^2}$ 来计算，这里 Δ_x，Δ_y，Δ_z 分别是在 X，Y，Z 轴上的误差分量值。

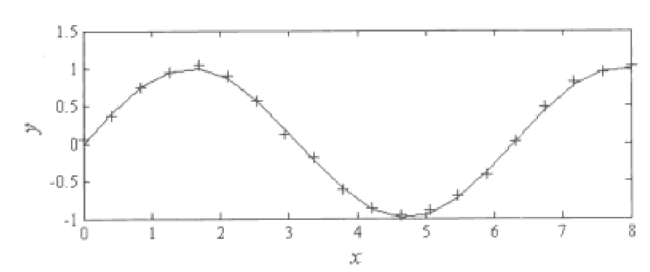


图 9.1　二维曲线的实验数据与拟合数据图示化比较

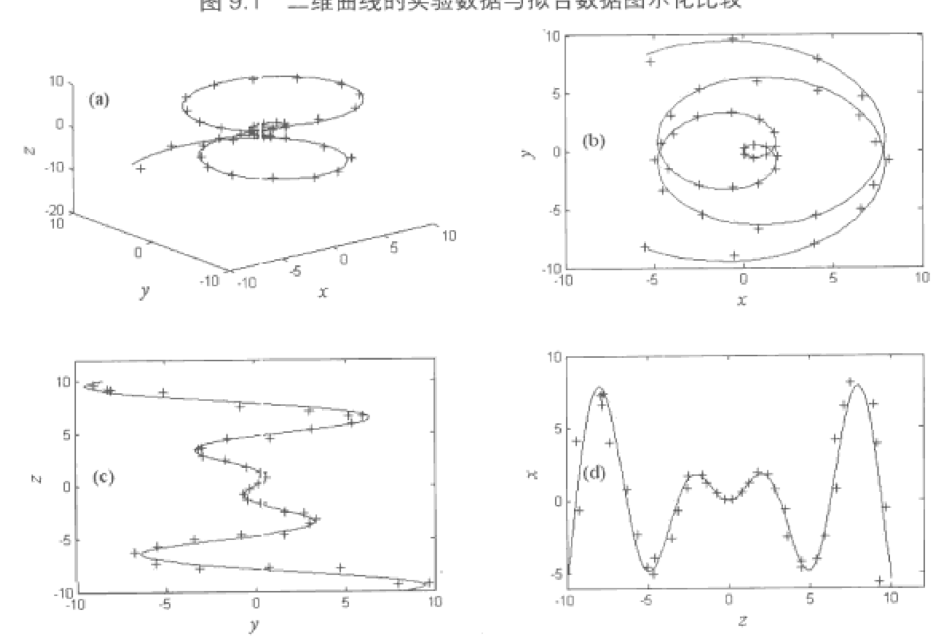


图 9.2　三维曲线实验数据与拟合数据图示化比较

对于形如 $y=f\left(x_1,x_2\right)$ 的数据拟合模型，可以绘制误差面来表示，如图 9.3 所示。

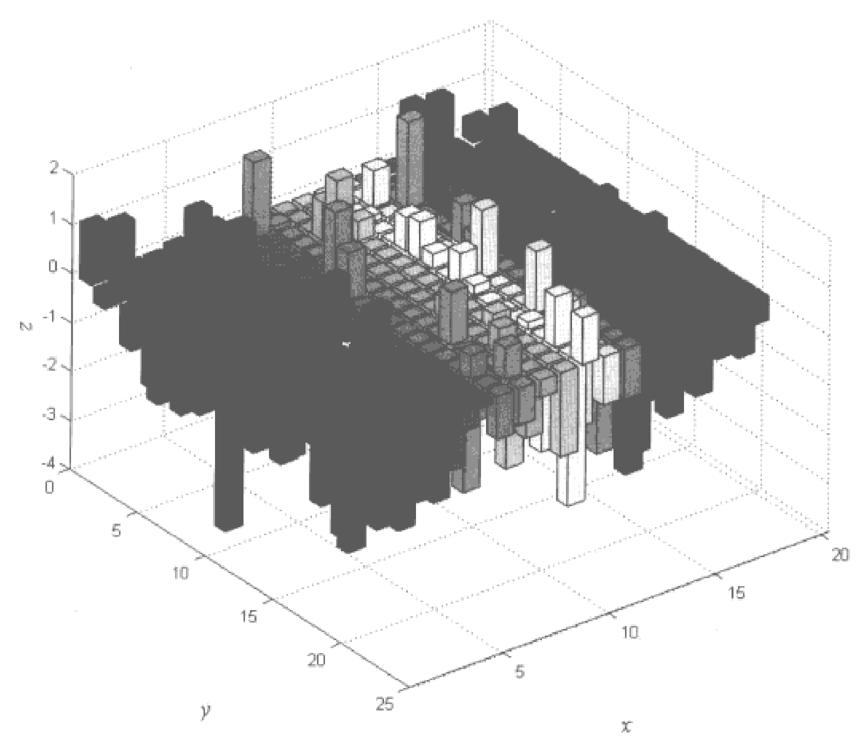


图 9.3 三维曲面对象的拟合误差比较

如图 9.3 所示，曲面各点值的计算公式是：$\Delta\left(x_k,y_k\right)=f\left(x_k,y_k\right)-z_k$，其中 z_k 是实验数据。

由此可见，利用图形来观察实验数据和拟合数据之间的误差非常方便、直观。在实际应用中，可以根据需要，选择不同的方法来查看拟合数据和实验数据之间的误差。

9.2 最小二乘拟合

逼近实验数据的基本方法就是拟合方法，其中最常用的就是最小二乘法拟合。最小二乘法适用的数学模型可以是线性函数和非线性函数。对于一组给定的数据 $\left(x_k,y_k\right)$，设计和确定一个函数模型 $y=f\left(x\right)$，使得函数 $f\left(x\right)$ 在某种误差函数标准下与左右数据点 $\left(x_k,y_k\right)$ 之间的距离最近，也就是说寻求一个最佳拟合函数。

最小二乘拟合是解决曲线拟合问题的常用方法。这里以线性函数为例介绍最小二乘法的原理，该方法的基本思路是建立一个线性模型，即 $y=f\left(x\right)=ax+b$。

其中 a 和 b 是待定系数。通过实验数据 $\left(x_k,y_k\right)$ 确定出系数 k 和 b。最小误差原则是保证 y_k 和 $f\left(x_k\right)$ 之间距离的平方和最小，因此这种方法被称为线性最小二乘法。

下面考虑系数 a 和 b 的求法。考虑下面的误差函数：

$$\Delta(a,b)=\sum_{k=1}^{n}\left[y_k-f(x_k)\right]^2=\sum_{k=1}^{n}\left[y_k-(ax_k+b)\right]^2$$

可以发现$\Delta(a,b)$是关于a和b的二次函数。根据高等数学的知识，$\Delta(a,b)$具有最小值的必要条件是$\frac{\partial\Delta(k,b)}{\partial k}=0$和$\frac{\partial\Delta(k,b)}{\partial b}=0$，从而有：

$$\begin{cases}\sum_{k=1}^{n}x_k\left[ax_k+b-y_k\right]=0\\ \sum_{k=1}^{n}\left[ax_k+b-y_k\right]=0\end{cases}$$

求解上面关于a和b的方程组有：$a=\frac{\overline{x_k y_k}-\overline{x}_k\overline{y}_k}{\overline{x_k^2}-\overline{x}_k^{\,2}}$，$b=\overline{y}_k-a\overline{x}_k$。其中$\overline{x}_k$和$\overline{y}_k$分别为$x_k$和$y_k$的均值，$\overline{x_k y_k}$和$\overline{x_k^2}$分别为$x_k y_k$和$x_k^2$的均值。

相应的误差评估通常使用下面两种形式。

◆ 最小平方根误差（均方误差）：$\Delta_1=\left\{\sum_{k=1}^{n}\left[y_k-f(x_k)\right]^2\right\}^{1/2}$

◆ 最大偏差：$\Delta_2=\max\limits_{1\le k\le}\left|y_k-f(x_k)\right|$

类似地对于多项式函数和更复杂的函数表示都可以利用上面的思路来确定模型中的未知系数。对于单个数学模型，使用最小二乘法只能获得一个拟合结果，可以计算相应的误差值，如果误差值没有达到要求，就要更改为其他模型，直到满足要求为止。

实际中，一些模型可以转化为线性最小二乘问题，如：

◆ $y=ax^m+b$，其中$m\neq1$，通过代换$u=x^m$，可以得到一个线性模型。

◆ $y=a\cdot e^{bx^m}$，通过代换$u=x^m$及两边取对数操作，可以简化为线性模型。

在实际应用中，需要考虑不同的模型，通过误差函数值的比较从中选择拟合后误差最小的。而函数模型的设计需要根据经验来选择，在数据规律不明显的时候，可以通过平移、取对数或者分段处理等操作来探索数据的规律。

例 9-1：计算线性最小二乘拟合，处理数据如表 9.1 所示。

表 9.1 线性最小二乘拟合数据

xk	0	1	2	3	4	5	6	7	8	9
yk	2	3.4	5.6	8	11	12.3	13.8	16	18.8	19.9

本问题程序如下：

```
xk = 0:9;
yk = [2 3.4 5.6 8 11 12.3 13.8 16 18.8 19.9];
% xk和yk为离散数据
```

```
plot(xdata,ydata,'r+')          % 画离散数据点
h=lsline                        % 最小二乘拟合
xd = get(h, 'XData');           % 获取直线的数据
yd = get(h, 'YData');           % 获取直线的数据
a = [yd(1)-yd(2)]/[xd(1)-xd(2)]  % 斜率
b = yd(1)-a*xd(1)               % 截距
```

a 和 b 的值如下：

```
a =    2.0582
b =    1.8182
```



lsline 的作用是在图形上添加一个线性最小二乘拟合的直线。在使用 lsline 函数之前，坐标轴上的数据点的曲线不能使用 3 种线型（属性 LineStyle），即实线、虚线和点划线。默认情况下，直线的颜色与数据点的颜色相同，而且返回句柄 h 中包含了如图 9.4 所示那条直线的全部信息。其中 h 的参数“XData”和“YData”是该直线的两个端点的坐标，根据两点法可以求出斜率和截距，从而确定直线方程 $y = ax + b$。

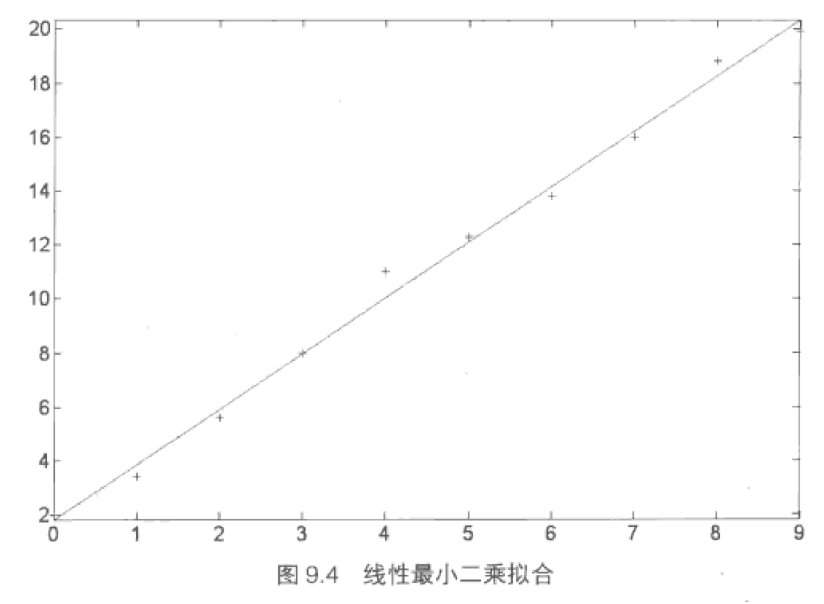


图 9.4 线性最小二乘拟合

此外函数 polyfit 可以用来进行多项式拟合，如果选择 1 次多项式，那么正好对应着上面的模型。为了比较上面的结果，这里调用此函数进行拟合。

```
P = polyfit(xk,yk,1)
```

结果为：

```
P =    2.0582    1.8182
```

这里 P(1)是 1 次项系数，P(2)是多项式的常数项。可以发现 polyfit 函数与 lsline 函数返回了相同的结果，利用 polyfit 函数可以方便地对实验数据进行线性最小二乘拟合处理。关于 polyfit 函数详细的使用方法将在下一节介绍。

接下来考虑一种多元线性拟合问题。这里以二元问题为例，一般的函数模型可以写为 $z=f(x,y)=ax+by+c$，其中 a，b 和 c 为待定系数。数据点 (x_k,y_k,z_k) 带入上述函数模型可以得到方程组 $ax_k+by_k+c=z_k$，（$1\le k\le n$）。

上面的方程组可以写为矩阵形式：$M\cdot X=d$，其中 $M=\begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & 1 \end{bmatrix}$，$Y=\begin{bmatrix} a \\ b \\ c \end{bmatrix}$，$d=\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix}$。

方程 $M\cdot X=d$ 可以转化为求下面的最小二乘优化问题：

$$\min_x \quad \frac{1}{2}\|MX-d\|_2^2$$

MATLAB 中提供了函数 lsqlin 计算最小二乘优化问题，其调用格式为：

```
[X, resnorm, residual] = lsqlin(M,d);
```

参数说明： X 是最小二乘解。resnorm 是 2-范数，resnorm=norm(C*x-d) ^2。residual 是残差，residual=C*x-d。M 和 d 分别描述方程 $M\cdot X=d$ 的矩阵和向量。

例 9-2： 计算下面的二元线性拟合问题，如表 9.2 所示。

表 9.2　二元线性拟合

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
y	1.8	1.7	1.6	1.5	1.4	1.2	1.0	0.85	0.67
z	5.1	5.4	5.7	6.0	6.3	6.4	6.5	6.7	6.8



上述数据在光盘中由 P0527.mat 保存。

相应程序如下：

```
load P0527                          % 读入数据
M=[x',y',ones(9,1)];                     % 组成 M 矩阵
d=z;                               % 生成向量 d
[X, resnorm, residual] = lsqlin(M,d)   % 计算最小二乘问题
X=X', resnorm, residual=residual'    % 显示结果
```

上述程序输出如下结果：

```
X =    5.1110    2.0964    0.8146
resnorm =  7.6085e-004
residual=  -0.0009   0.0006   0.0021   0.0035   0.0050  -0.0032  -0.0114  -0.0147
0.0190
```

这说明 $a=5.1110$，$b=2.0964$，$c=0.8146$，同时拟合的数据和实验数据之间的误差比较小，其中最大误差是 0.0190。

有的时候在测量数据中存在着一些明显误差点，也称它们为坏点。在进行拟合计算时，不希望考虑这些点。MATLAB 提供了函数 robustfit 来完成健壮性稳健回归分析，该函数具体调用格式为：

```
B = robustfit(x, y);
[B, STATS] = robustfit(x, y);
[B, STATS] = robustfit(x, y, 'wfun', tune, 'const');
```

参数说明：B 返回系数估计向量。STATS 返回各种参数估计。wfun 指定一个加权函数，它可以选取下面的名字：andrews，bisquare，cauchy，fair，huber，logistic，talwar，welsch。tune 为调协常数。const 的值为 on（其为默认值）时添加一个常数项，为 off 时忽略常数项。

例 9-3：处理带有一个异常点的数据，比较最小二乘拟合值与稳健拟合差异。

分析：首先利用 $y=8+3x$ 加上正态随机噪音项生成模拟数据，然后改变最后一个 y 值产生异常值。调用不同的拟合函数（lsline，polyfit 和 robustfit），通过图形和数据输出观察拟合函数的效果。程序如下：

```
randn('state',2007);                    % 设置随机数状态
x = [1:8]';                             % 产生 x 数据
y = 8+3*x+randn(8,1);                   % 产生 y 数据
y(end)=2;                               % 设置最后一个数据为异常数据
plot(x,y,'o','MarkerFaceColor','k');    % 画出数据点
h=lsline;                               % 添加最小二乘结果
set(h,'lineStyle',':','Color','r');     % 改变线型和颜色
P=polyfit(x,y,1);                       % 一次多项式拟合
B=robustfit(x,y);                       % 稳健拟合
hold on
plot(x,B(1)+B(2)*x,'k');                % 绘制稳健拟合曲线
set(gca,'Fontsize',16);                 % 字体设置
xlim([min(x),max(x)]);                  % 设置 x 轴范围
set(gcf,'color','w');                   % 设置背景色
P, B                                    % 输出直线系数到命令窗
```

输入图形如图 9.5 所示，P 和 B 的值如下：

```
P =    0.5016   15.6436
B =    8.1030
3.0180
```

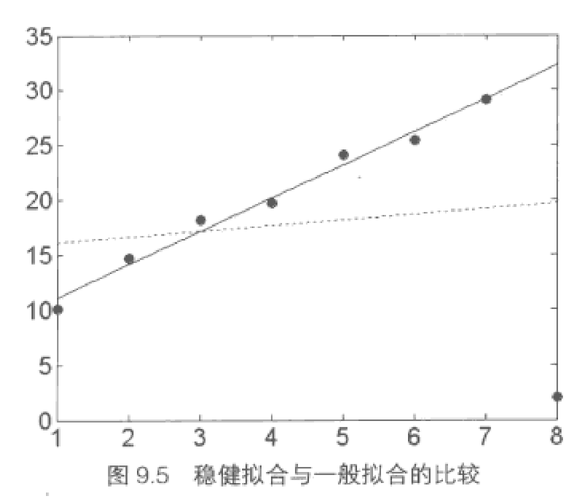


图 9.5　稳健拟合与一般拟合的比较

可见函数 robustfit 没有考虑最后那个坏点，对数据的拟合程度好些，而函数 lsline 和 polyfit 把所有数据点都考虑进去进行最小二乘计算，拟合结果受到异常值影响，直线偏向异常值一侧。在实

际应用中可以根据不同需要选择拟合方式。

9.3 多项式拟合

这里考虑那些无法通过数学代换转为线性模型的拟合问题。令 $f(x,a)$ 表示一个非线性函数，其中 a 表示待定的系数组，即 $a=[a_1,a_2,...,a_n]$。我们知道任何连续函数都可以利用幂函数来展开，这样就可以得到一个多项式函数了。本节将介绍多项式拟合，其中系数可以通过最小二乘法来计算。

MATLAB 中的 polyfit 函数可以完成多项式拟合，其调用格式为：

```
P = polyfit (X, Y, N);
[P, S] = polyfit (X, Y, N);
```

参数说明： 其中 P 表示多项式系数，它对应于多项式的降幂顺序，即 Y = P(1)*X^N + P(2)*X^(N-1) + … + P(N)*X + P(N+1)。S 可以用来误差估计和预测数据结构体。X 和 Y 是输入的实验数据，N 表示拟合计算用到多项式的次数。此外，polyfit 函数还存在下面一种调用格式"[P, S, MU] = polyfit (X, Y, N);"，该语句完成对归一化处理数据 XHAT=(X-MU(1))/MU(2)的多项式拟合计算，MU 是由均值和标准方差构成的 1×2 向量，即 MU=[mean(X),std(X)]。

例 9-4： 多项式拟合。

```
x = 1:6;                              % 数据 x
y = [3,9,18,32,50,72];                    % 数据 y
[P1, S, MU] = polyfit(x,y,2);               % 对数据 x 归一化处理的多项式拟合
P1
P2 = polyfit([x-mean(x)]/std(x),y,2)  % 等价形式的多项式拟合
P3 = POLYFIT(x, y, 2)               % 无归一化处理多项式拟合
```

输出如下：

```
P1 =    7.2500   25.7640   24.6250
P2 =    7.2500   25.7640   24.6250
P3 =    2.0714   -0.7286    1.8000
```

使用"[P, S, MU] = polyfit (X, Y, N);"的时候需要特别注意，它与"P = polyfit(X, Y, N);"得到的结果差异很大。

与 polyfit 函数配合使用的函数是 polyval，这个函数根据拟合出来的多项式系数 P 计算给定数据 X 处的 Y 值。其调用格式如下：

```
Y = polyval(P, X);
[Y, DELTA] = polyval(P, X, S);
[Y, DELTA] = polyval(P, X, S, MU);
```

参数说明： Y 是根据多项式系数 P 计算出来的 X 处的多项式值。DELTA 是利用结构体 S 计算出来的误差估计，Y 的 95%置信区间[Y－DELTA,Y+DELTA]，其中假设 polyfit 函数数据输入的误差是独立正态的，并且方差为常数。函数 polyval 的输入参数与函数 polyfit 的输出参数意义相同。

为了更好地分析拟合结果，可以使用函数 polyconf，该函数可以给出多项式拟合结果评价和置

信区间，其调用格式为：

```
[Y,DELTA]=polyconf(P,X,S);
[Y,DELTA]=polyconf(P,X,S,ALPHA);
```

参数说明： Y，DELTA，P，X，S 意义同函数 polyval，参数 ALPHA 与置信度有关。同样，函数 polyconf 也假设函数 polyfit 的输入数据误差是独立正态的，并且方差为常数。

例 9-5： 计算下面数据的 2 到 6 次多项式拟合结果，如表 9.3 所示。

表 9.3 多项式拟合数据

x	-4	-3	-2	-1	0	1	2	3	4
y	2.2	-3.2	-2.1	0.16	1.0	0.04	-0.66	3.4	19

分析： 为了同时得到不同次数多项式拟合的系数，这里借助了一个循环结构来计算不同次数的多项式拟合，其中使用了 for 循环结构，把拟合系数保存到一个矩阵中。程序如下：

```
x = -4:4;                    % 数据 x
y = [2.2, -3.2, -2.1, 0.16, 1.0, 0.04, -0.66, 3.4, 19];    % 数据 y
P = zeros(5,7);               % 初始化多项式系数矩阵
for k=2:6;
   [Pt,S] = polyfit(x,y,k);       % 多项式拟合
   P(k-1,1:k+1) = Pt;          % 把 Pt 赋值给系数矩阵 P
   Y = polyval(Pt,x,S);         % 计算 x 处的拟合值
   Dt(k-1) = std(Y-y);          % 计算拟合值与原数据之间的标准方差
end
P,Dt                         % 显示拟合系数矩阵和误差
```

输出的系数矩阵和误差如下：

```
P =
   0.6425    1.4960   -2.0790         0         0         0         0
   0.1442    0.6425   -0.2061   -2.0790         0         0         0
   0.0999    0.1442   -0.9982   -0.2061    1.0028         0         0
  -0.0002    0.0999    0.1490   -0.9982   -0.2244    1.0028         0
   0.0001   -0.0002    0.0979    0.1490   -0.9870   -0.2244    0.9939
Dt =     3.3232    2.7085    0.0183    0.0140    0.0121
```

可见当多项式次数大于 3 时误差迅速减小，同时随着次数的增加误差减小，而且次数较高的时候高次项系数变得很小，因此对于这组数据使用 4 次多项式拟合即可。对于不能确定多项式次数的时候，建议读者尝试不同次数的多项式拟合，对结果的误差进行分析，从而选择合适的多项式次数。



数据 x 的单调性不会影响拟合结果。用下面的 Ik 向量打乱 x 和 y 的顺序，再进行拟合。

```
Ik = [1, 7, 4, 3, 6, 2, 9, 8, 5];          % 扰乱顺序
x = x(Ik);                         % 重排 x 数据
y = y(Ik);                         % 重排 y 数据
```

得到下面的结果：

```
P =
    0.6425    1.4960   -2.0790         0         0         0         0
    0.1442    0.6425   -0.2061   -2.0790         0         0         0
    0.0999    0.1442   -0.9982   -0.2061    1.0028         0         0
   -0.0002    0.0999    0.1490   -0.9982   -0.2244    1.0028         0
    0.0001   -0.0002    0.0979    0.1490   -0.9870   -0.2244    0.9939
Dt =    3.3232    2.7085    0.0183    0.0140    0.0121
```

可以发现上面的结果与 x 按递增顺序排列的结果完全一样，因此使用者在输入数据时保证 x 和 y 的对应关系即可。

例 9–6： 利用从 2 到 5 次多项式拟合下面的数据，并在 figure 上动态显示结果，如表 9.4 所示。

表 9.4　动态显示结果

x	0	1	2	3	4	5
y	1	22	61	69	76	111

分析：用 text 函数及 title 函数分别显示多项式系数和次数。程序如下：

```
x = 0:5;                    % 输入拟合 x 数据
y = [1 22 61 69 76 111];      % 输入拟合 y 数据
xp = 0:0.02:5;               % 产生更密集的 x 轴数据
for N=2:5
    P = polyfit(x,y,N);      % N 次多项式拟合
    yp = polyval(P,xp);      % 计算 xp 的多项式值
    plot(x,y,'r+',xp,yp,'k');   % 绘制数据点与拟合曲线
    title(['n=',int2str(N)],'Fontsize',16);  % 显示次数
    for k=1:length(P);
       text(1+k*0.4,k*10,['P(',num2str(k),...
              ')=',char(vpa(P(k),4))],...
              'FOntsize',16);     %显示所有多项式系数
    end
    set(gca,'Fontsize',14);            %设置字体大小
    set(gcf,'Color','w');             % 重置背景色
    pause                       % 停顿，按任意键继续循环计算
end
```

按空格键可以得到从 2 到 5 次的多项式拟合结果，截图如图 9.6 所示。这样就可以实时观察拟合结果了，读者可以套用上面的程序进行多项式拟合分析。

函数的最佳多项式逼近科学计算中常用的方法是，在一个有限的区间（如[0,1]）上，寻求简单函数的逼近方法。前面提到使用函数 polyfit 可以得到 N 次拟合多项式的系数。如果已知一个较为复杂的连续函数 $f(x), x\in[x_1,x_2]$，求一个较为简单的函数 $S(x)$，如多项式函数 $p(x)$，则用最小二乘法来逼近函数 $f(x)$，就是函数逼近。它不同于前面的数据逼近。

与实验数据拟合类似，函数逼近的误差使用积分代替求和，其定义如下：

$$\Delta(x)=\int_{x_1}^{x_2}\left[S(x)-f(x)\right]^2 dx$$

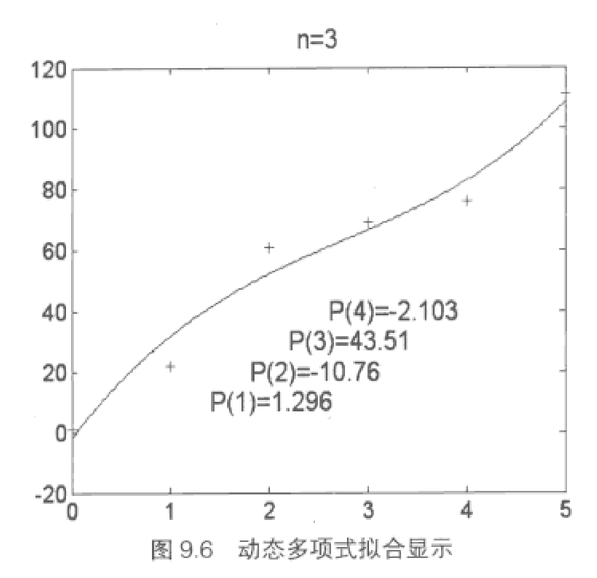


图 9.6　动态多项式拟合显示

如果误差函数$\Delta(x)$达到最小，那么函数$S(x)$就是$f(x)$的一个最佳逼近。这里首先定义一组基函数$\left[b_1(x),b_2(x),...,b_n(x)\right]$，逼近函数$S(x)$可以表示为这些基函数的线性叠加：

$$S(x)=c_1b_1(x)+c_2b_2(x)+\cdots+c_nb_n(x)$$

其中$c_1,c_2,...,c_n$为待定系数。为了求$\Delta(x)$的最小值，可以得到下面的方程组：

$$\begin{bmatrix} I_{1,1} & \cdots & I_{1,n} \\ \vdots & \vdots & \vdots \\ I_{n,1} & \cdots & I_{n,n} \end{bmatrix}\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}=\begin{bmatrix} Q_1 \\ Q_2 \\ \vdots \\ Q_n \end{bmatrix}$$

其中$I_{i,j}$和Q_i（$1\leqslant i$，$j\leqslant$n）表示内积，即$I_{i,j}=\int_{x_1}^{x_2}b_i(x)b_j(x)\mathrm{d}x$，$Q_i=\int_{x_1}^{x_2}f(x)b_i(x)dx$。

根据线性方程组知识，上述方程组具有唯一解的充要条件是系数矩阵非奇异。

在实际应用中，最简单的基函数就是幂函数，即$b_1(x)=1$，$b_2(x)=x$，$b_3(x)=x^2$，等等。同时如果所选基函数能满足$I_{i,j}=\delta_{i,j}=\begin{cases}1, & i=j \\ 0, & i\neq j\end{cases}$，就是说系数矩阵是对角矩阵，这样上述方程组的求解将极大地被化简。这样的多项式被称为正交多项式，显然幂函数不能满足这个性质。

满足上面正交关系的多项式有勒让德（Legendre）多项式、拉盖尔（Laguerre）多项式和第一类切比雪夫（Chebyshev）多项式等。它们的定义分别如下。

◆ 勒让德多项式：$q_0(x)=1$，$q_n(x)=\dfrac{1}{2^n n!}\dfrac{\mathrm{d}^n}{\mathrm{d}x^n}\left(x^2-1\right)^n$, $(n=1,2,3,..)$

◆ 拉盖尔多项式：$L_n(x)=e^x\dfrac{\mathrm{d}^n}{\mathrm{d}x^n}\left(x^n e^{-x}\right)$，$\left(x\in[0,+\infty),\ n=0,1,2,...\right)$

◆ 第一类切比雪夫多项式：$T_n(x)=\cos(n\arccos x)$，$\left(x\in[-1,1],\ n=0,1,2,...\right)$

例 9-7： 计算函数$f(x)=\sin x,\ x\in[0,\ \pi]$在基函数组$\left\{x,x^3,x^5\right\}$上的逼近多项式系数$c_1$，$c_2$，$c_3$。

分析：计算系数矩阵 I 和向量 Q 的各元素值，然后通过解方程组就可以得到系数 c_1，c_2，c_3 的数值。相应程序如下：

```
syms x                           % 定义符号变量
order = [1,3,5]';                % 幂函数次数列向量
[m,n] = meshgrid(order);         % 次数网格矩阵
Ix = x.^[m+n];                   % 系数矩阵的被积函数
Qx = sin(x)*x.^order;            % 向量中的被积函数
Iv = double(int(Ix,x,0,pi));     % 计算积分确定矩阵元素值
Qv = double(int(Qx,x,0,pi));     % 计算积分确定向量元素值
C = Iv\Qv                        % 计算待定系数并输出到命令窗
fx = taylor(sin(x),6)            %  sin(x)的 5 次麦克劳林多项式
```

上述程序输出下面的结果：

```
C =
    0.9879
   -0.1553
    0.0056
fx =x-1/6*x^3+1/120*x^5
```

通过比较可以得出系数 C 与 $\sin x$ 的泰勒级数展开吻合。

9.4 非线性拟合

MATLAB 提供了两个求非线性最小二乘拟合函数，即 lsqcurvefit 和 lsqnonlin。函数 lsqcurvefit 的调用格式如下：

```
X = lsqcurvefit('fun',X0,XDATA,YDATA);
```

参数说明：X 是拟合输出的系数，fun 是非线性函数模型，它的定义要用 M 文件定义一个函数文件。X0 是 X 的初始值。XDATA 是自变量的数据，YDATA 是因变量的数据。该函数在旧版本中的对应名称是 curvefit。

函数 lsqnonlin 的调用格式如下：

```
X = lsqnonlin ('fun', X0);
```

参数说明：X 是拟合输出的系数，fun 是一个误差函数，等于因变量值和用非线性函数模型计算在因变量处的值的差，它的定义也需要使用 M 文件。X0 是 X 的初始值。该函数在旧版本中的对应名称是 leastsq。

例 9-8：使用非线性函数模型。

用函数 $y=f\left(x\right)=c_1+c_2e^{-0.02c_3x}$（其中 c_1，c_2，c_3 是待定系数）来拟合下面的数据，如图 9.7 和表 9.5 所示。

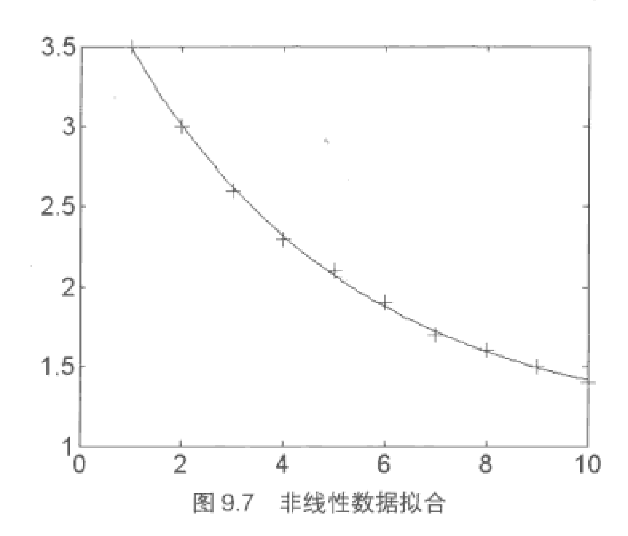


图 9.7 非线性数据拟合

表 9.5 拟合数据

x	1	2	3	4	5	6	7	8	9	10
y	3.5	3.0	2.6	2.3	2.1	1.9	1.7	1.6	1.5	1.4



上表的数据存储在光盘中的 P0528.mat 文件中。

分析：这个函数模型是不能简化为线性函数的，因此考虑进行非线性拟合。这里考虑使用 lsqcurvefit 函数与 lsqnonlin 函数分别拟合。

应用 lsqcurvefit 函数时，首先要定义函数模型，根据表达式得到下面的函数文件：

```
function y=fit_model_a(x,xd);       % 定义 lsqcurvefit 拟合函数的输入函数
y=x(1)+x(2)*exp(-0.02*x(3)*xd);     % x(1)-->c_1,x(2)-->c_2,x(3)-->c_3
load P0528                     % 读入数据
c0 = [0,1,1];                     % 初始值
c = lsqcurvefit('fit_model_a',c0,xd,yd)  % 进行非线性拟合
plot(xd,yd,'r+');                    % 利用原始数据绘图
hold on;
xp = linspace(min(xd),max(xd),200);  % 生成原始数据范围内的等间距采样点
yp = fit_model_a(c,xp);  % 利用拟合系数计算因变量数值
plot(xp,yp);                       % 绘制拟合曲线
```

输出拟合结果为：

```
c =   1.0979    2.9918   11.2336
```

接下来使用函数 lsqnonlin 拟合上面的数据，此时 fun 的定义应该是下面的格式。

```
function y=fit_model_b(x);              % 定义 lsqnonlin 拟合函数的输入函数
load P0528                        % 读入数据
y=yd-[x(1)+x(2)*exp(-0.02*x(3)*xd)];      % x(1)-->c_1,x(2)-->c_2,x(3)-->c_3
```

主程序书写如下：

```
c0 = [0,1,1];                          % 初始值
```

```
c = lsqnonlin('fit_model_b',c0)          % 进行非线性拟合
```

输出如下：

```
c =1.0979    2.9918    11.2336
```

可见两个函数的输出结果是一样的，但是它们拟合函数的输入方式不同。此外初始值 x0 的选取也会影响结果，可以套用上面的格式来解决自己的问题。

多元非线性拟合问题可以利用多元非线性回归分析，这部分内容将在第 11 章讲述。

9.5 Lagrange 插值

在实际应用中，插值问题的数据可能是一维或者二维。首先我们考虑一维插值问题。一维插值问题的数学描述是：函数 $y=f(x)$ 在区间 $[a,b]$ 上有 N 个数据点 (x_k,y_k) 已知，此外函数 $f(x)$ 的某些阶导数是已知的，通过插值计算确定区间 $[a,b]$ 上任意一点 x 的函数值。

一维插值的基本思路是：根据函数 $f(x)$ 在区间 $[a,b]$ 上的已知数据点 (x_k,y_k)，求解出一个足够光滑、简单的函数 $g(x)$ 作为函数 $f(x)$ 的近似表达式，其中 $g(x)$ 被称为插值函数。两个函数之间的关系是：$g(x_k)=f(x_k)=y_k$，（$1\le k\le N$）。

接下来计算函数 $g(x)$ 在区间 $[a,b]$（它也叫做插值区间）上其他点 x 的函数值为函数 $f(x)$ 的近似值。多项式函数是最简单的函数，它常作为插值基函数。

假设 $g(x)$ 是一个满足条件的多项式，同时它的最高次数不超过 $N-1$，它可以写为下面的形式：

$$g(x)=a_N x^{N-1}+a_{N-1}x^{N-2}+\cdots+a_2x+a_1$$

其中 $a_1,a_2,\ldots,a_N$ 是待定系数，带入已知的数据点 (x_k,y_k) 可以得到下面的方程组：

$$\begin{cases} a_N x_1^{N-1}+a_{N-1}x_1^{N-2}+\cdots+a_2x_1+a_1=y_1 \\ a_N x_2^{N-1}+a_{N-1}x_2^{N-2}+\cdots+a_2x_2+a_1=y_2 \\ \qquad\cdots \\ a_N x_N^{N-1}+a_{N-1}x_N^{N-2}+\cdots+a_2x_N+a_1=y_N \end{cases}$$

由 x_k 组成的系数矩阵记为 A，矩阵 A 的行列式正好是范德蒙（Vandermonde）行列式。当 x_k 彼此不相等时，该行列式不等于 0，此时方程组有唯一解。所有系数确定之后，插值多项式就计算出来了。相应的插值多项式 $g(x)$ 与函数 $f(x)$ 之间的差值称为截断误差或者是插值余项，即 $R(x)=f(x)-g(x)$，（$x\in[a,b]$）。

显然 $x=x_k$ 时，$R(x)=0$。

本节将给出拉格朗日（Lagrange）插值方法，拉格朗日插值的基函数是：

$$l_k = \frac{(x-x_1)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_N)}{(x_k-x_1)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x_k-x_N)} = \prod_{n\neq k}\frac{x-x_n}{x_k-x_n}$$

显然，上面的基函数是一个分式函数的连乘积。各多项式的次数都是 $N-1$，同时满足下面的关系式：

$$l_k(x_j) = \begin{cases} 0, & j\neq k \\ 1, & j=k \end{cases}，（1\leq j,k\leq N）$$

利用 $l_k(x)$ 可以构造出下面的插值函数 $g_L(x)$：

$$g_L(x) = \sum_{k=1}^{N} y_k l_k(x)$$

显然 $g_L(x)$ 是一个 $N-1$ 次多项式，同时满足插值条件：

$$g_L(x_k) = y_k$$

可见拉格朗日插值的系数求解根据上面的公式即可得出，不需要复杂的运算。

当原函数 $f(x)$ 在区间 $[a,b]$ 上比较光滑时，根据洛尔（Rolle）定理可以推导出对于任意 $x\in[a,b]$，插值余项是：

$$R(x) = f(x) - g_L(x) = \frac{f^N(\xi)}{N!}\omega_N(x),\ \xi\in[a,b]$$

$$\omega_N(x) = \prod_{k=1}^{N}(x-x_k)$$

如果 $f^N(\xi)$ 有界，即 $f^N(\xi)\leq M$，可以得到拉格朗日插值的余项是：

$$|R(x)| \leq \frac{M}{N!}|\omega_N(x)|$$

在 MATLAB 中没有专门的函数来计算拉格朗日插值。Carlo Castoldi 编写了拟合拉格朗日多项式插值程序，用户可以在 MathWorks 公司网站下载：http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=899&objectType=file。

直接地以拉格朗日插值公式计算需要用到三重循环，还需要判断语句。这里给出一种只用一重循环的程序。首先构造矩阵 W，Wk，X，Xk，即：

$$W = \begin{bmatrix} x_1 & x_2 & \cdots & x_N \\ x_1 & x_2 & \cdots & x_N \\ \vdots & \vdots & \vdots & \vdots \\ x_1 & x_2 & \cdots & x_N \end{bmatrix}；\ Wk = x_k - \begin{bmatrix} x_1 & x_2 & \cdots & x_N \\ x_1 & x_2 & \cdots & x_N \\ \vdots & \vdots & \vdots & \vdots \\ x_1 & x_2 & \cdots & x_N \end{bmatrix}$$

，其中 Wk 的第 k 行所有元素等于 1；

$$X = \begin{bmatrix} x \\ x \\ \vdots \\ x \end{bmatrix} - \begin{bmatrix} x_1 & x_1 & \cdots & x_1 \\ x_2 & x_2 & \cdots & x_2 \\ \vdots & \vdots & \cdots & \vdots \\ x_N & x_N & \cdots & x_N \end{bmatrix}；\ Xk(m,n) = \begin{cases} X(m,n), m\neq k \\ y_k, \quad m=k \end{cases}$$

，即 Xk 的第 k 行等于 y_k，其他行与 X 对应行相等。

其中W和Wk的大小是$N\times N$，X和Xk的大小是$N\times L$（L是离散数据x的长度）。然后对于前面提到的每一个插值基函数$l_k(x)$，计算出矩阵Wk和Xk的点除运算，即"$Wk./Xk$"。再对结果矩阵的各列求连乘积（利用 prod 函数），从而得到$y_k l_k(x)$。相应程序如下：

```
function y=lag_interp(x,xd,yd);
%Example:
%x=1:.1:4;                 % 待求点数据
%xd=1:4;                  % 插值点自变量数据
%yd=[2,1,3,4];              % 插值点因变量数据
%y=lag_interp(x,xd,yd);    % 计算拉格朗日插值
%plot(x,y,'k',xd,yd,'ko','markersize',10);  %绘制曲线
N = length(xd);              % 插值点自变量数据长度
L = length(x);               % 待求点数据长度
W = repmat(xd',1,L);         % 生成矩阵 W
X = repmat(x,N,1)-repmat(xd',1,L);   % 矩阵 Xt
y = 0;                       % 初始化 y
for k = 1:N;
   Xk = X;
   Xk(k,:) = yd(k)*ones(1,L);  % 生成矩阵 Xk
   Wk = xd(k)-W;
   Wk(k,:) = 1;              % 生成矩阵 Wk
   y = y+prod(Xk./Wk);       % 累积求和
end
```

参数说明： 参数 x 是待求点坐标，xd 和 yd 是插值点坐标数据。这里要求 x，xd，yd 都是行向量。y 是待求点的因变量值。

执行上述函数的例子可以得到如图 9.8 所示的结果，其中圆圈表示插值数据点，插值得到的曲线经过插值数据点，与前面的结论一致。

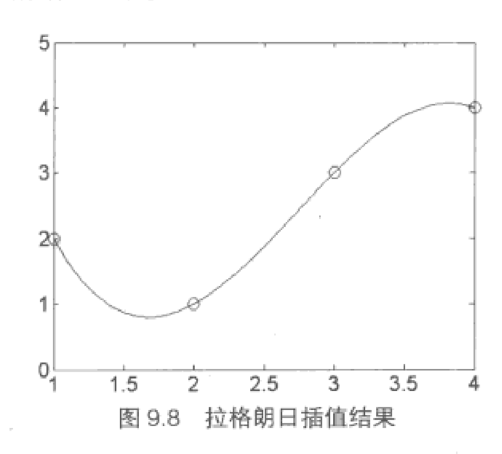


图 9.8　拉格朗日插值结果

9.6 Hermite 插值

本节来介绍另一种插值方法——厄尔米特（Hermite）插值。厄尔米特插值公式如下：

$$y=g_H(x)=\sum_{k=1}^{N}h_k\left[(x_k-x)(2a_k y_k-y_k')+y_k\right]$$

其中 $h_k=\prod_{j\neq k}\left(\frac{x-x_j}{x_k-x_j}\right)^2$，$a_k=\sum_{j\neq k}\frac{1}{x_k-x_j}$，（$1\le k,j\le N$），$y_k'$ 是一阶导数值。

可见厄尔米特插值表达式比拉格朗日插值要复杂得多。其中多项式的最高次数是 $2N-1$，因此厄尔米特插值的光滑性比拉格朗日插值要好。

与拉格朗日插值编写类似，利用一重循环根据厄尔米特插值公式编写如下程序：

```
function y=hermite_interp(x,xd,yd,dy);
% Hermite 插值
N = length(xd);                 % 计算插值点数据长度
L = length(x);                  % 计算待求点数据长度
X = repmat(x,N-1,1);            % 扩展为矩阵 X
y = 0;                          % 初始化 y
for k = 1:N;
    xt = xd;
    xt(k) = [];                 % 去掉不满足条件的求和求积项
    h=prod(1./[xd(k)-xt]);
    h=h*prod([X-repmat(xt',1,L)]);
    h = h.^2;                   % 计算 h 乘积因子
    a = sum(1./[xd(k)-xt]);     % 计算 a 乘积因子
y = y+h.*[(xd(k)-x)*(2*a*yd(k)-dy(k))+yd(k)];  % 对 y 累加
end
```

参数说明：参数 x 是待求点坐标，xd 和 yd 是插值点坐标，dy 是对应一阶导数值。这里要求 x，xd，yd，dy 都是行向量。y 是待求点的因变量值。

利用下面这段程序，调用上面的函数文件 hermite_interp.m 进行厄尔米特插值：

```
x=1:.1:4;                                   % 待求点数据
xd=1:4;                                     % 插值点自变量数据
yd=[2,1,3,4];                               % 插值点因变量数据
rand('state',0);                            % 设置随机数状态
dy = rand(1,4)-0.5;                         % 产生随机的 dy 值
y=hermite_interp(x,xd,yd,dy);               % 插值计算
plot(x,y,'k',xd,yd,'ko','markersize',10);   % 绘图
```

上述程序执行结果如图 9.9 所示。与拉格朗日插值不同的是，厄尔米特插值需要已知一阶导数值。

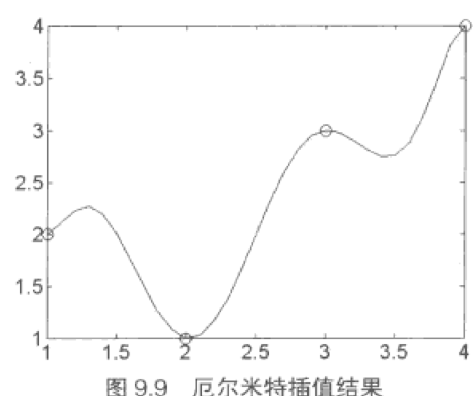


图 9.9 厄尔米特插值结果

9.7 样条插值

利用拉格朗日插值和厄尔米特插值的时候遇到高次多项式插值时会出现收敛性和稳定性变差现象，结果不理想，导致龙格（Runge）现象。该现象最早是 Runge 在 19 世纪初发现的，他计算了区间 $[-1,1]$ 上的 N 个点等间距插值，使插值多项式在各个插值点与函数 $y=1/(1+25x^2)$ 相等。在 N 较大时，插值多项式在区间中部趋于函数 $y=1/(1+25x^2)$，但是在区间两侧，即 $|x|\geq 0.726$，插值多项式发生严重发散。

下面用拉格朗日插值和厄尔米特插值计算这个插值问题。主要程序如下（略去绘图部分）：

```
N = 5;                              % 设定采样点数
xd = linspace(-1,1,N);              % 产生插值点
yd = 1./(1+25*xd.^2);               % 产生插值点
x = linspace(-1,1,200);             % 获得待求点坐标
y = 1./(1+25*x.^2);                 % 正确的结果
yL = lag_interp(x,xd,yd);           % 拉格朗日插值
rand('state',0);                    % 设定随机数状态
dy = rand(1,N)-0.5;                 % 产生随机的一阶导数
yH = hermite_interp(x,xd,yd,dy);    % 进行厄尔米特插值
```



光盘中的完整程序名是 runge_test.m。

图 9.10 中的实线是根据函数 $y=1/(1+25x^2)$ 表达式计算的曲线，虚线是插值结果，圆圈表示插值点。可以发现采样点数等于 13 时，两种插值多项式在区间两侧的结果明显偏离正确结果。中间部分与正确值比较吻合。因此，如果函数表达式未知，是无法确定使用多少插值数据点是合适的。

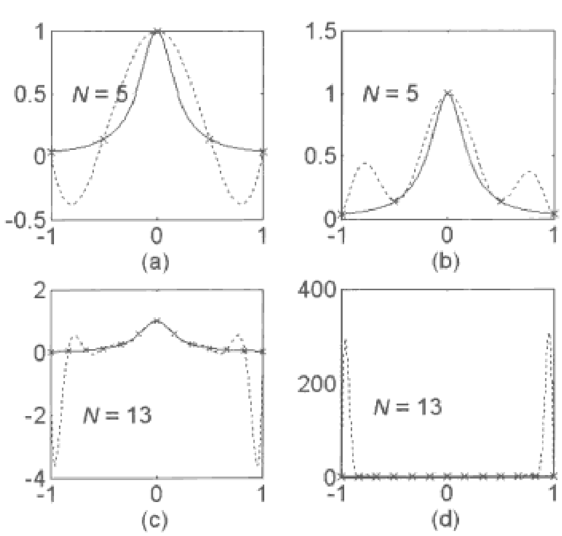


图 9.10　龙格现象：(a)和(c)是拉格朗日插值，(b)和(d)是厄尔米特插值

上述实例说明使用高次多项式插值是很危险的，因此在实际应用中应尽量避免使用高次多项式插值。这就启发我们使用少的插值数据来完成插值计算，从而对于大量数据而言，就产生了分段插

值的想法。把区间$[a,b]$划分为多个子区间，在每个子区间内进行分段的低次多项式插值。自区间的划分方法可以根据具体问题的要求选择。分段插值具有较好的全域收敛性和稳定性，而且算法简单，但缺点是插值函数的光滑性差。常用的分段插值有两类：一类是分段线性插值，另一类是样条插值。

分段线性插值的做法是：认为区间$[a,b]$上的连续函数$f(x)$在N个节点（$a=x_1<x_2<\cdots<x_N=b$）上的函数值是$y_k=f(x_k)$，（$1\le k\le N$）。依次连接相邻的插值点可以得到$N-1$条线段组成的一条折线。把这条折线称为被插函数$f(x)$的分段线性插值函数，记为$S(x)$，可以证明函数$S(x)$满足下面 3 个性质：

- $S(x)$可以分段表示，在每个子区间$[x_{k-1},x_k]$上，它是线性函数，即：
$$S(x)=y_{k-1}\frac{x-x_k}{x_{k-1}-x_k}+y_k\frac{x-x_{k-1}}{x_k-x_{k-1}},\ x\in[x_{k-1},x_k]$$
- 满足插值条件$S(x_k)=y_k$，$1\le k\le N$。
- $S(x)$在区间$[a,b]$上，但是不能保证处处可导。

如果定义如下基函数：

$$L_k(x)=\begin{cases}\dfrac{x-x_{k-1}}{x_k-x_{k-1}}, & x\in[x_{k-1},x_k] \text{ and } 1<k\le N\\ \dfrac{x-x_{k+1}}{x_k-x_{k+1}}, & x\in[x_k,x_{k+1}] \text{ and } 1\le k<N\\ 0, & \text{otherwise}\end{cases}$$

基于这些基函数$S(x)$可以表示为$S(x)=\sum_{k=1}^{N}y_kL_k(x)$。

当函数$f(x)$在区间$[a,b]$上连续时，分段线性插值函数$S(x)$具有很好的收敛性，即$\lim_{x\to 0}S(x)=g(x)$，$x\in[a,b]$，$h=\max_{1<k\le N}\{x_k-x_{k-1}\}$（最大子区间长度）。

上述关系可以根据微积分基本知识证明。同时当$f(x)$在区间$[a,b]$上有二次连续导数时，对$\forall x\in[a,b]$有插值余项关系$|R(x)|=|f(x)-S(x)|\le\frac{M}{8}h^2$，$M=\max_{x\in[a,b]}\{|g''(x)|\}$。

用分段插值函数$S(x)$计算区间$[a,b]$上x的插值结果时，只是要用到包含x的子区间端点值，与其他的节点无关，而且计算过程就是通过平面上两点坐标求直线表达式的方法。根据插值余项可知，N越大，即h越小，插值误差越小。实际中用数据点进行插值计算时，使用分段线性插值就可以满足要求了，如数学和物理中用到的特殊函数数值表、概率论与数理统计中的概率分布函数表都是使用分段线性得到的。

MATLAB 提供了专门计算分段线性插值的函数 interp1，该函数的完整调用格式为：

```
yi = interp1(x, y, xi, 'method' ,'extrap');
yi = interp1(x,y,xi, 'method',extrapval);
```

参数说明： x 和 y 是插值点，x 是待求点坐标，y 是待求点因变量值。参数 method 可以选择的值有：nearest 表示最近邻插值，linear 表示线性插值（其为默认值），spline 表示三次样条函数插值，pchip 表示分段三次 Hermite 插值（同时 pchip 自身也是一个完成分段三次 Hermite 插值的 MATLAB 函数），cubic 表示立方插值，v5cubic 表示在 MATLAB 5.0 中的三次插值。其中对于超出 x 范围的 xi 分量，使用 nearest，linear 和 v5cubic 的插值算法，将返回 NaN。

对其他的方法，interp1 将对超出的分量执行外插值算法。对于超出 x 范围的 x(k)中的分量将执行特殊的外插值法 extrap。参数 extrapval 的意思是确定超出 x 范围的 xi 分量其外插点的因变量值等于 extrapval，其值用户可以自行定义。经过试验发现，x 和 y 的大小排列顺序不影响插值结果，只要保证 x 和 y 个数据准确对应即可。

例 9–9： 用不同的插值方法计算函数 $f(x)=x^2\cos x$ 在区间 $[0,5]$ 上的分段插值。

具体程序如下：

```
x = 0:.5:5;                 % 生成插值点自变量数据
% N = length(x);             % 获得插值数据长度
% Ik = randperm(N);          % 产生一个随机排序序列
% x = x(Ik);                 % 对 x 随机排序
y = x.^2.*cos(x);                                    % 计算插值点因变量数值
xi = 0:.02:5;                                        % 计算待求点自变量数据
% xi = fliplr(xi);                                     % 左右翻转向量 xi
method = {'nearest','linear','spline','pchip','cubic','v5cubic'};    % 插值方法名数
组
for k = 1:6;
   subplot(3,2,k);                                   % 分布子图顺序
   yi = interp1(x,y,xi,char(method(k)));        % 用不同方法分段插值
   yg = xi.^2.*cos(xi);                             % 计算理论值
   plot(x,y,'ko',xi,yi,'k:',xi,yg,'k');                  % 画数据点及绘制曲线
   text(2,5,char(method(k)),'Fontsize',16);     % 标注插值方法
   set(gca,'Fontsize',14);                              % 重置坐标轴字体
   xlim([min(x),max(x)]);                            % 设定横轴范围
end
```

程序输出结果如图 9.11 所示。其中虚线是插值结果，实线是理论曲线，圆圈表示插值点。因为插值方法是 nearest，对于 x 点，其插值结果取据 x 最近的插值点的因变量数值，所以该函数得到的结果在变化大的地方看着像台阶。分段线性插值是一条由线段组成的折线。另外 4 种插值方法得到的结果是比较光滑的曲线。



从图 9.11 可以发现选用 spline 和 v5cubic 方法进行插值时，插值结果和理论结果重合得非常好。此外上面的程序注释掉的语句可以测试 x 和 y 数据随机排列时对插值结果的影响，以及 xi 数据左右翻转后对结果的影响。用户可以自己取消注释进行观察。

前面介绍的分段线性插值函数，在节点处的一阶导数一般情况下是不存在的，且光滑性不高，这些限制了该插值方法在诸如机械设计等领域（即希望插值曲线光滑）中的应用。许多工程技术中提出的计算问题对插值函数的光滑性有较高要求，如飞机的机翼外形，内燃机的进、排气门的凸轮曲线等，都要求曲线具有较高的光滑度，不仅要连续，而且要有连续的曲率。在船舶、飞机零件等

设计中，对于已知的一些数据点(x_k, y_k)，机械设计人员的一般做法是：先将这些数据点画在平面图纸上，再用一根富有弹性的弯曲细直条（称为样条），使其一边通过这些数据点，用压铁固定细直条的形状，沿样条边沿绘出一条光滑曲线。实际需要用几根样条线分段完成上述工作，如此可以保证相邻样条在连接处也保持光滑。

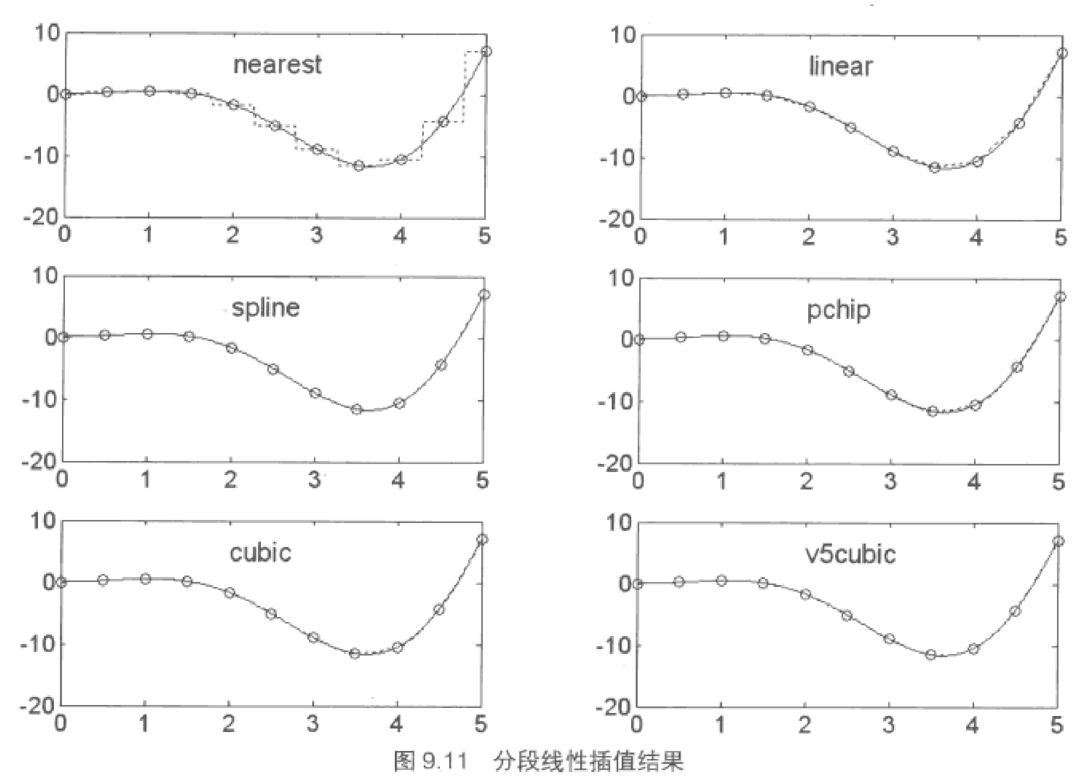


图 9.11　分段线性插值结果

根据力学知识，如此画出的曲线在相邻两数据点之间实际上是次数不高于 3 的多项式，在整个区间上有连续的曲率。对机械设计人员用样条画出的曲线进行数学抽象，从而导出了样条插值函数的概念。定义在区间$[a,b]$上的分段函数$S(x)$，如果满足下面的条件：

- $S(x)$在各个子区间$[x_{k-1}, x_k]$上都是 3 次多项式。
- $S(x)$在整个区间$[a,b]$上存在连续的二阶导数。

那么函数$S(x)$就是区间$[a,b]$上的一个 3 次样条插值函数。类似地，可以定义 n 次样条插值函数。样条函数的主要优点是它的光滑程度较高，保证了插值函数二阶导数的连续性，对于三阶导数的间断，人类的眼睛已难以辨认了。样条函数是一种隐式格式，最后需要解一个方程组，它的工作量大于多项式拉格朗日插值或者厄尔米特插值等显式插值方法。

在插值计算之前，N 个插值点是已知的。在每个子区间上有 4 个待定参数（即 3 次多项式的系数），因此未知参数的数目是$4N$。而在每个子区间上，根据插值条件可以由两端点得到两个方程，这样得到的方程数目是$2N$个。此外根据光滑性，在相邻的两个子区间的共同端点处，一阶和二阶导数连续，可以得到$2(N-1)$个方程。至此得到方程总数目是$4N-2$。为了保证参数的唯

一性，还需要再找到两个方程。常用的方法是对整个区间的端点除函数值外再附加要求，这就是所谓的边界条件。根据实际问题的不同，3 次样条插值常用到下列 3 类边界条件。

◆ 给定一阶导数：$S'(a)=y_1'$，$S'(b)=y_N'$，即预先给定a和b处的一阶导数值。由这种边界条件建立的样条插值函数称为$f(x)$的完备三次样条插值函数。当$y_1'=y_N'=0$时，样条曲线在端点处呈现水平状态。如果y_1'和y_N'不确定，还可以认为$S'(x)$在端点处y_1'和y_N'近似相等，通过两端 8 点，用x_1,x_2,x_3,x_4和$x_{N-3},x_{N-2},x_{N-1},x_N$分别进行 3 次拉格朗日插值得到$L_a(x)$和$L_b(x)$。令$S'(a)=L_a'(a)$和$S'(b)=L_b'(b)$。

用拉格朗日插值得到一阶导数的方法称为拉格朗日三次样条插值。

◆ 给定二阶导数：$S''(a)=y_1''$，$S''(b)=y_N''$。当$y_1''=y_N''=0$时，这样的边界条件被称为自然边界条件。

◆ 周期性边界条件：当函数$f(x)$是一个周期为$b-a$的周期函数时，自然地可以认为$S(x)$也是一个周期为$b-a$的周期函数。因此两端点处的一阶和二阶导数相等，即$S'(a)=S'(b)$和$S''(a)=S''(b)$。

上述 3 种边界条件都可以得到两个方程，加上前面提到的$4N-2$个方程，就可以得到$4N$个独立方程，来唯一地确定三次插值函数的$4N$未知参数。

如果三次样条插值函数没有边界条件，最常用的办法是采用非扭结（not a knot）条件，即设定第 1 个和第 2 个 3 次多项式的三阶导数相等，同时设定最后两个 3 次多项式的三阶导数也相等。

在子区间$[x_{k-1},x_k]$上的三次样条插值函数表达式是：

$$S_k(x)=-M_{k-1}\frac{(x-x_k)^3}{6h}+M_k\frac{(x-x_{k-1})^3}{6h}-\left(y_{k-1}-\frac{h^2}{6}M_{k-1}\right)\frac{x-x_k}{h}+\left(y_k-\frac{h^2}{6}M_k\right)\frac{x-x_{k-1}}{h}$$

其中$h=x_k-x_{k-1}$，M_{k-1}和M_k分别为x_{k-1}和x_k处的 2 次导数（其表达式比较复杂，这里略去）。在子区间$[x_{k-1},x_k]$上的另外一种表达式是：

$$S(x)=y_{k-1}\left(1+2\frac{x-x_{k-1}}{h}\right)\frac{(x-x_k)^2}{h^2}+m_{k-1}(x-x_{k-1})\frac{(x-x_k)^2}{h^2}$$
$$+y_k\left(1+2\frac{x-x_k}{h}\right)\frac{(x-x_{k-1})^2}{h^2}+m_k(x-x_k)\frac{(x-x_{k-1})^2}{h^2}$$

其中$h=x_k-x_{k-1}$，m_{k-1}和m_k分别为x_{k-1}和x_k处的 1 次导数（其表达式比较复杂，这里略去）。

当函数$f(x)$存在 4 阶导数时，样条插值函数的误差估计是$\left|f^{(n)}(x)-S^{(n)}(x)\right|\le c_nMh^{4-n}, n=0,1,2, x\in[a,b]$，其中$c_0=5/384$，$c_1=1/24$，$c_2=3/8$，$M=\max\limits_{x\in[a,b]}\left\{\left|g^{(4)}(x)\right|\right\}$。

MATLAB 中提供了实现三次样条插值的函数，即 interp1，spline 和 csape，它们的调用格式如下：

```
yi = interp1(x, y, xi);
yy = spline(x, y, xx);
```

```
pp = csape (x, y, conds, valconds)
```

参数说明： x 和 y 是已知的插值点，x 是待求点的自变量值。函数 csape 是专门的三次样条插值函数，输出参数 pp 是一个结构体，需要调用函数 ppval 来计算各因变量的数值。

参数 conds 表示选用的插值边界条件，其默认的插值方法是拉格朗日边界条件，具体取值为：complete 或者 clamped 表示边界为一阶导数，端点的一阶导数值在参数 valconds 中给出，如果参数 valconds 缺省，此时就是拉格朗日边界条件；not-a-knot 表示非扭结边界条件；periodic 表示周期性边界条件；second 表示边界为二阶导数，导数值由参数 valconds 给出，如果参数 valconds 缺省，二阶导数的默认值是[0,0]，即自然边界条件。variational 设置边界二阶导数等于[0,0]（就是自然边界条件）。

对于一些特殊的边界条件，可以通过参数 conds 取一个 1×2 的向量来表示，其中元素取值为 0，1，2。conds(i)=j 表示设置为 j 阶导数，该向量的第一个值对应于左边界，第二个值对应于右边界。相应导数值由 valconds 给出。利用函数 csape 可以解决不同边界条件的插值问题。

例 9–10： 利用 csape 函数对下面的数据进行样条插值计算，如表 9.6 所示。

表 9.6　样条插值计算数据

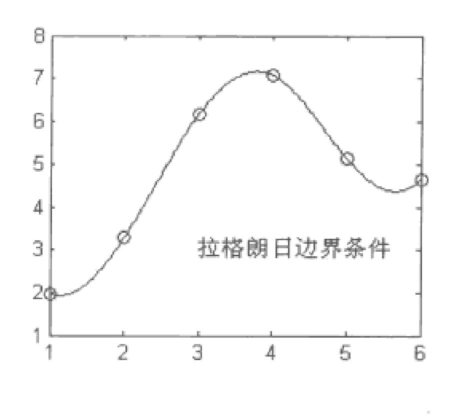
x	1	2	3	4	5	6
y	-0.48	1.94	3.29	6.16	7.07	7.63

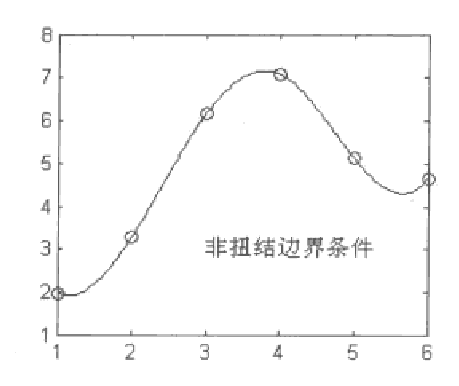
分析： 比较不同的边界条件进行插值的计算。程序如下：

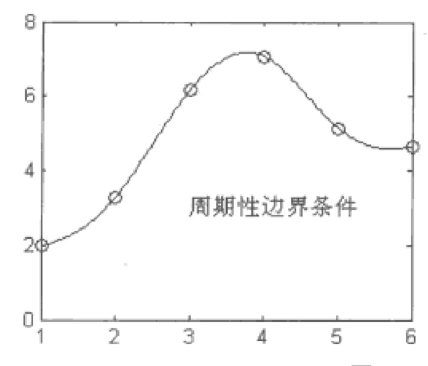
```
x = 1:6;
y = [1.98 3.28 6.16 7.07 5.14 4.66];
xx = linspace(min(x),max(x),200);
subplot(221);
pp = csape(x, y, 'comlete');        % 拉格朗日边界条件
yy = ppval(pp,xx);               % 计算插值曲线数据
plot(x,y,'ko',xx,yy,'k');           % 绘图
subplot(222);
pp = csape(x, y, 'not-a-knot');      % 非扭结边界条件
yy = ppval(pp,xx);                % 计算插值曲线数据
plot(x,y,'ko',xx,yy,'k');           % 绘图
subplot(223);
pp = csape(x, y, 'periodic');        % 周期性边界条件
yy = ppval(pp,xx);                % 计算插值曲线数据
plot(x,y,'ko',xx,yy,'k');            % 绘图
subplot(224);
pp = csape(x, y, 'second');         % 自然边界条件
yy = ppval(pp,xx);                % 计算插值曲线数据
plot(x,y,'ko',xx,yy,'k');            % 绘图
```

图 9.12 显示了不同边界条件下的插值结果，我们发现几种边界条件的结果整体相似，但是细微的差别还是存在的，可以根据自己的问题来选择边界条件。

在 MATLAB 样条工具箱中提供了样条的建立、操作等功能。其中的样条函数，根据前缀可以分为 4 类：cs*是三次样条、pp*是分段多项式样条（系数为 t^n 的系数）、sp*是 B 样条（系数为基函数 B_n^i(t)的系数）、rp*是有理 B 样条。下面把相关函数整理为一个表格，如表 9.7 所示。







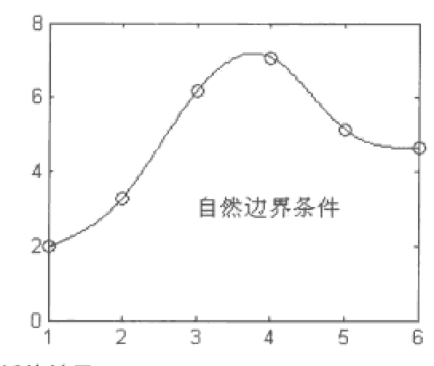


图 9.12　样条插值结果

表 9.7　样条函数表

分类	函数名	说明
三次样条函数	csape	生成给定约束条件下的三次样条函数
	csapi	插值生成三次样条函数
	csaps	平滑生成三次样条函数
	cscvn	生成一条内插参数的三次样条曲线
	getcurve	动态生成三次样条曲线
分段多项式样条函数	ppmak	生成分段多项式样条函数
	ppual	计算在给定点处的分段多项式样条函数值
	ppval	计算在给定点处的分段多项式样条函数值
B 样条函数	spap2	用最小二乘法拟合生成 B 样条函数
	spapi	插值生成 B 样条函数
	spaps	对生成的 B 样条曲线进行光滑处理
	spcol	生成 B 样条函数的配置矩阵
	spcrv	生成均匀划分的 B 样条函数
	spmak	生成 B 样条函数
有理样条函数	rpmak	生成有理样条函数
	rsmak	生成有理样条函数
	rsval	计算在给定点处的有理样条函数值
样条操作函数	fmbrk	返回样条函数的某一部分（如断点或系数等）
	fn2fm	把一种形式的样条函数转化成另一种形式的样条函数
	fncmb	对样条函数进行算术运算
	fnder	求样条函数的微分（即求导数）

（续表）

分类	函数名	说明
样条操作函数	fndir	求样条函数的方向导数
	fnint	求样条函数的积分
	fnjmp	在间断点处求函数值
	fnplt	画样条曲线图
	fnrfn	在样条曲线中插入断点
	fntlr	生成 tarylor 系数或 taylor 多项式
	fnval	计算在给定点处的样条函数值
样条曲线端点和节点处理函数	aptknt	求出用于生成样条曲线的节点数组
	augknt	在已知节点数组中添加一个或多个节点
	aveknt	求出节点数组元素的平均值
	brk2knt	增加节点数组中节点的重次
	chbpnt	求出用于生成样条曲线的合适节点数组
	knt2brk	从节点数组中求得节点及其重次
	knt2mlt	从节点数组中求得节点及其重次
	newknt	对分段多项式样条函数进行重分布
	optknt	求出用于内插的最优节点数组
	sorted	求出节点数组的元素在另一节点数组中属于第几个分量

拉格朗日插值函数在整个插值区间上有统一的解析表达式，其形式关于节点对称，光滑性好。但缺点同样明显，这主要体现在高次插值收敛性差（龙格现象）；增加节点时前期计算作废，导致计算量大；一个节点函数值的微小变化（观测误差存在）将导致整个区间上插值函数都发生改变，因而其稳定性差等。因此拉格朗日插值法多用于理论分析，在采用拉格朗日插值方法进行插值计算时，通常选取最高插值次数小于 7。厄尔米特插值方法的光滑性要高于拉格朗日插值，但是同样存在着龙格现象。因此这两种插值方法在实际中应用较少。

分段线性插值函数（仅连续）与三次样条插值函数（二阶导数连续）虽然光滑性差，但它们都克服了拉格朗日插值函数的缺点，不仅收敛性、稳定性强，而且方法简单实用，计算量小。因而它们的工程应用十分广泛。

这里补充使用快速 Fourier 算法做一维插值的方法。这里认为被插函数是周期函数，计算函数值的傅里叶变换，然后使用更多的点进行傅里叶变换的逆变换，函数的使用格式如下：

```
y=interpft(x,n)
```

其中 x 是已知函数值，要求是等距的点，n 为返回等间距点的个数（n 要求不小于 x 的长度）。

9.8 二维插值

前面讲的插值是一维插值，而实际中可能遇到自变量数据是两个自由度，即二维插值问题。二维插值问题的数学表述是：已知二元函数 $f(x,y)$ 在矩形区域（$x\in[a,b]$，$y\in[c,d]$）内系列点 (x_m, y_n) 的函数值 $z_{m,n}$，计算该矩形区域内任意一点的值近似函数 $f(x)$。

根据数据点 (x_m, y_n) 分布的情况，常见的二维插值问题可分为两种情况：网格节点插值和散乱节点插值。其中网格节点插值适用于节点比较规范的情况，即在包含所给节点的矩形区域内，节

点由两组平行于坐标轴的直线的交点组成。散乱节点插值适用于一般的节点，多用于节点不太规范（即节点为两组平行于坐标轴的直线的部分交点）的情况。下面分别介绍这两种情况的插值方法。

9.8.1 网格节点插值法

常见的插值方法有下面 3 种方式。

◆ 分片线性插值

分片线性插值对应于分段线形插值，思想是把考虑区域分割为多个子矩形，即以(x_m, y_n)，(x_{m+1}, y_n)，(x_{m+1}, y_{n+1})和(x_m, y_{n+1})为顶点的小矩形$r_{m,n}$上进行线性插值，相应的分片插值函数是：

当$x\in[x_m, z_{m+1}]$，且$y\in[y_n, k(x-x_m)+y_n]$时，

$g(x,y)=z_{m,n}+(z_{m+1,n}-z_{m,n})(x-x_m)+(z_{m+1,n+1}-z_{m+1,n})(y-y_n)$；

当$x\in[x_m, z_{m+1}]$，且$y\in[k(x-x_m)+y_n, y_{n+1}]$时，

$g(x,y)=z_{m,n}+(z_{m+1,n+1}-z_{m,n+1})(x-x_m)+(z_{m+1,n+1}-z_{m,n})(y-y_n)$。

其中$k=[y_{n+1}-y_n]/[x_{m+1}-x_m]$，即以直线$y=k(x-x_m)+y_n$分割矩形为两部分，分别用不同的线性函数插值。可见分片线性插值函数$g(x,y)$连续，但光滑性不好。

◆ 分片双线性插值

分片双线性插值曲面由一片片空间二次曲面构成，其分片函数表达式是：

$$g(x,y)=(Ax+B)(Cy+D),\ (x,y)\in r_{m,n}$$

其中A，B，C和D是待定系数，它们可以根据小矩形的 4 个顶点唯一确定，然而在相邻小矩形的 4 条边（不含定点）上不能保证连续。

◆ 分片双三次样条插值

分片双三次样条插值函数在各个小矩形$r_{m,n}$上的表达式如下：

$$g(x,y)=(A_1+A_2x+A_3x^2+A_4x^3)(B_1+B_2y+B_3y^2+B_4y^3)$$

待定系数A_k，B_k（$k=1,2,3,4$）可以由子矩形 4 个顶点的z值及插值函数$g(x,y)$在x和y方向的光滑性（即偏导数g'_x，g'_y，g''_{xx}和g'_{yy}连续）和相应的边界条件唯一确定。

9.8.2 散乱节点插值

常用的方法是反距离加权平均法（或者称为 Shepeard 法）。其基本思路是，在点(x,y)处，定义其插值函数的函数值为点(x_m, y_n)的函数值$z_{m,n}$，按(x,y)与(x_m, y_n)之间的距离的某种形式反比例作为权重因子的加权平均。如：

$$g(x,y)=\begin{cases} z_{m,n}, & r_{m,n}=0 \\ \sum\limits_{m,n} W_{m,n}(x,y)\,z_{m,n}, & r_{m,n}\neq 0 \end{cases}$$

其中 $r_{m,n}=\sqrt{(x-x_m)^2+(y-y_n)^2}$，$W_{m,n}(x,y)=\frac{1}{r_{m,n}^2}\sum_{m,n}\frac{1}{r_{m,n}^2}$，这样定义的插值函数是与全局相关的，对曲面 $g(x,y)$ 上任一点做数值计算都要涉及到全体已知数据，因而在已知数据量大的情况下计算量相当大，而且插值曲面在节点 (x_m,y_n) 处不光滑。即使如此，由于该方法思想简单，在此基础上还可以有种种改进方法。

MATLAB 中提供了函数 interp2 和 csape 来实现二维插值。函数 interp2 的调用格式如下：

```
zi = interp2(x, y, z, xi, yi, 'method');
```

参数说明： zi 是返回的插值矩阵。x 和 y 是长度分别为 M 和 N 的向量，对应着已知点 (x_m,y_n)。z 是一个矩阵，对应于 $z_{m,n}$。method 的用法和意义同函数 interp1。

如果进行三次样条插值，可以使用函数 csape，其使用方法是：

```
pp = csape({x,y},z,{'method1','method2'}, condsval);
```

参数说明： pp 是一个结构体，其中包含样条函数的系数。x 和 y 是向量，表示已知点。z 是已知点的函数值。method1 和 method2 可以分别对 x 和 y 方向设置边界条件，具体含义可以参见前面一维插值的介绍。

例 9-11： 在一次对砂堆形状测量的时候得到部分高度信息如表 9.8 所示。

表 9.8 砂堆形状测量的高度信息

单位：米		x			
		1	2	3	4
y	1	6.36	6.97	6.23	4.77
	2	6.98	7.12	6.31	4.78
	3	6.83	6.73	5.99	4.12
	4	6.61	6.25	5.53	3.34

利用二维插值计算该区域内其他点的数值。

分析： 这里分别使用函数 interp2 和函数 csape 进行插值，这里把 z 数据保存为 sands.mat 文件。程序如下：

```
x = 1:4;                        %生成插值点数据 x
y = 1:4;                        %生成插值点数据 y
load sands;                     %读入数据
xi = linspace(1,4,20);                  % 生成待求点 X 轴数据
yi = xi;                              % 生成待求点 Y 轴数据
[xx,yy]=meshgrid(xi);                  % 生成坐标网格数据
z1 = interp2(x,y,z,xx,yy,'linear');          % 线性插值
pp = csape({x,y},z);                   % 样条插值
z2 = ppual(pp,{xi,yi});                   % 计算样条插值结果
subplot(121);mesh(xi,yi,z1);view([-8,42]); % 绘图，线性插值结果对应于图 9.13 的左图
subplot(122);mesh(xi,yi,z2);view([-8,42]);  % 绘图，样条插值结果对应于图 9.13 的右图
```

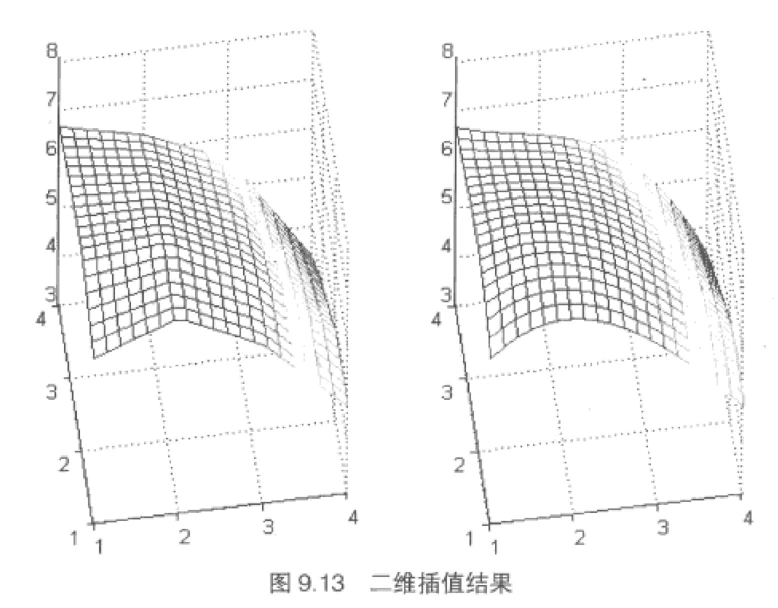


图 9.13 二维插值结果

可见样条插值得到的是比较光滑的曲面，而线性插值则是有折痕的表面（其由多个平面组成）。函数 griddata 可以用来做不规则的数据向量 x，y，z 的二元插值问题。其调用格式为

```
zi = griddata (x, y, z, xi, yi ,'method');
```

函数 griddata 将返回插值曲面 $z = g(x, y)$ 在点(xi, yi)处的插值。曲面总是经过这些数据点(x, y, z)的。输入参量(xi, yi)通常是规则的格点（像用命令 meshgrid 生成的一样）。参数 method 可以选择的值有：linear 是基于三角形的线性插值（默认方法）、cubic 是基于三角形的三次插值、nearest 是最邻近插值法、v4 是 MATLAB4 中的 griddata 算法。

类似地可以使用 interp3 进行三维插值，使用 interpn 和 ndgrid 进行 n 维插值。它们的使用方法与 interp1 和 griddata 相似，读者可以参阅帮助文档。

9.9 小结

本章主要介绍了数据拟合和插值，它们在数据分析时具有重要作用。首先介绍了数据拟合中的误差分析方法。通过最小二乘法，可以建立线性和非线性的数据拟合算法，其中经常用到的是多项式拟合。而非线性拟合可以用来解决复杂的函数模型。在介绍插值计算的时候，首先给出了两种经典的插值方法，即拉格朗日插值和厄尔米特插值，它们可以得到较为光滑的插值函数，但是在多项式次数较高的时候存在数据不收敛的问题，即龙格现象。使用分段插值可以解决这个插值不发散的问题，其中分段线性插值和分段样条插值具有较小的计算量，同时收敛效果好，特别是样条插值可以得到较光滑的函数，因此在实际中应用广泛。最后介绍了二维插值和高维插值函数。

◆ ◆ ◆

第 10 章　最值问题的求解

本章包括

- 极值的计算方法
- 离散和连续情况的最值求解
- 线的绘制

在数据分析、线性规划和优化等问题中经常要用到最值、极值的求解，此外如等式的求解也可以转化为模的最小值求解。本章结合多种涉及到最值求解的实际问题进行程序化计算，来介绍 MATLAB 在最值求解方面的应用。在 MATLAB 中提供了优化工具箱来解决最值问题的相关函数，读者可以在路径 MATLAB65\toolbox\optim 中查阅相关信息。

10.1　极值计算

极值是在某个定义范围函数的最大值或最小值。问题的对象可以有连续和离散两种情况，需要使用不同的方法对它们求解。本节分别介绍连续和离散情况下的求解方法。

10.1.1　连续情况

在数学上，极值的必要条件是函数 $f(x)$ 在 x_0 处的一阶导数等于 0，即：

$$\left.\frac{\mathrm{d}f(x)}{\mathrm{d}x}\right|_{x=x_0}=0 \tag{10-1}$$

极大值和极小值的区分是根据函数 $f(x)$ 在 x_0 处二阶导数取值的正负情况确定，即：

极大值条件：$$\left.\frac{\mathrm{d}^2 f(x)}{\mathrm{d}x^2}\right|_{x=x_0}<0 \tag{10-2}$$

极小值条件：$$\left.\frac{\mathrm{d}^2 f(x)}{\mathrm{d}x^2}\right|_{x=x_0}>0 \tag{10-3}$$

在 MATLAB 中，可以使用 diff 函数求连续函数的一阶和二阶导数，函数调用格式如下：

```
diff (S)
diff (S, 'v')
diff (S, n)
diff (S, 'v', n)
```

```
diff (S, n, 'v')
```

参数说明：其中 S 是定义的符号函数，它可以通过函数 sym 和 syms 定义符号变量来定义，或者函数的字符串表达式来定义（如 S ='sin(x)'），但是不支持 inline 函数定义的函数关系。v 是变量名称。n 是求导的次数。

下面通过一个求极值的例子来说明上述方法的程序实现。

10.1.1.1 一元函数的极值

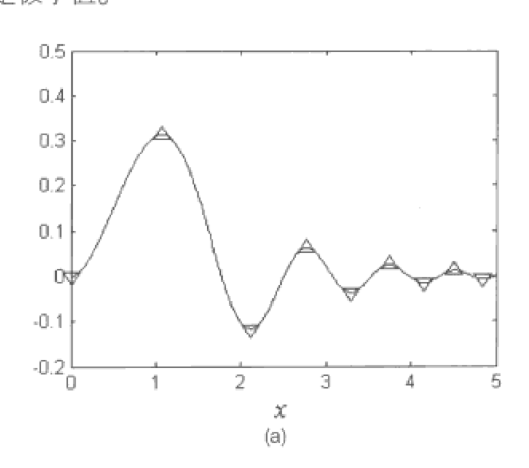
一元函数求极值的方法是先计算一阶导数的表达式，再计算一阶导数的零点，这些零点就是极值的位置点。当零点的二阶导数是负数时对应极大值，反之是极小值。

求函数 $f(x)=e^{-x}\sin x^2$ 在[0,5]区间上的极大值和极小值。

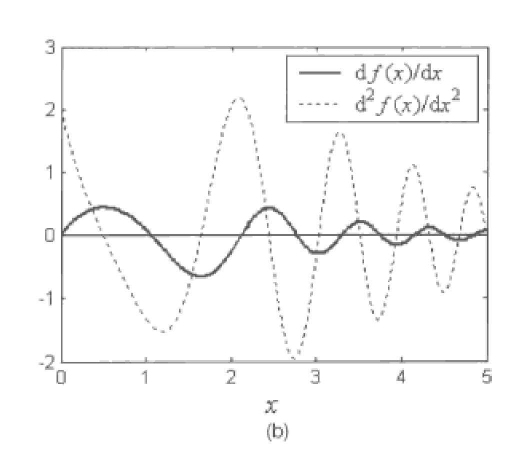
图 10.1(a)给出函数 $f(x)$ 在区间[0,5]的曲线，首先计算这个连续函数的一阶导数和二阶导数，相关程序如下：

```
syms x;                         % 定义符号变量 x
Sw = exp(-x)*sin(x^2);          % 定义函数表达式
dS=diff(Sw);                    % 计算一阶导数
d2S=diff(Sw,2);                 % 计算二阶导数
```

上述程序中，两条曲线 x~dS 和 x~d2S 由图 10.1(b)给出，其中粗实线表示一阶导数，可以发现该曲线多次穿过 0 刻度线。从二次导数的数值（由虚线表示）可以读出对应位置属于极大值还是极小值。



(a)



(b)

图 10.1 一元函数极值问题

现在来考虑如何利用程序实现一阶导数 0 点的求解。已经知道利用函数 fzero 和 fsolve 可以求出方程的根，但是这两个函数都是计算给定初值附近的根，无法直接计算整个区间内所有的根。这里将对 fzero 函数做一个扩展，编写了 fzeros.m 程序来计算方程在整个区间的根。扩展思路如下：在整个区间上等间距地获得多个取样点，即 $x_1, x_2, \cdots, x_n$，然后计算出相应的函数值 $f(x_1), f(x_2), \cdots, f(x_n)$。

对各点逐一判断，计算函数 $F(k)=f(x_{k-1})f(x_k)$，$2\le k\le n$ 是否非正。如果 $F(k)\le 0$，那么就用求根函数 fzero 计算 x_k 附近的根。这样通过多次调用 fzero 函数就可以得到区间上所有的

根。用下面的程序可以算出所有的极大值和极小值。

```
dS=char(dS);                    % 把 dS 的类型从符号型变为字符串型
dS=strrep(dS,'*','.*');              % 用点乘 ".*" 代替 dS 中的乘 "*"
dS=strrep(dS,'^','.^');              % 用运算 ".^" 代替 dS 中的乘 "^"
x0=fzeros(inline(dS),[0,5]);    % 用 fzeros 函数计算区间[0, 5]上的根
d2F=subs(d2S,x0);                % 计算方程所有根处的二次导数
xM=x0(d2F<0)                    % 从 x0 中分离出极大值点
xm=x0(d2F>0)                    % 从 x0 中分离出极小值点
```



在 fzero 函数的输入参数中，要求函数表达式是支持矩阵型运算的字符串型数据，因此需要把幂次和乘除（"^"、"*"、"/"）运算转换为相应的点运算（".^"、".*"、"./"）。为此，上面的程序中用 char 函数先获得字符串数据，用 strrep 函数进行运算类型替换（因为这里不含除运算，相应转换未进行）。

程序输出如下：

```
xM =    1.0637    2.7705    3.7422    4.5066
xm =         0    2.1167    3.2932    4.1423    4.8435
```

程序计算出的极大值位置 xM 和极小值位置 xm 分别用符号 "Δ" 和 "▽" 在图 10.1(a)中标出，可以看出所有极大值和极小值被正确地求出。本问题完整的程序名称是 min_maxs1.m。

10.1.1.2 二元函数的极值

与一阶导数的求解类似，计算二阶导数需要同时考虑两个自变量方向的一阶导数，当它们同时为零时，也就对应着极值点。极大值还是极小值可以通过判断零点与周围邻近点的大小关系。计算下面函数在 $x, y \in [-3, 3]$ 的极值：

$$z = f(x, y) = 3(1-x)^2 e^{-x^2-(y+1)^2} - 10\left(\frac{x}{5} - x^3 - y^5\right)e^{-x^2-y^2} - \frac{1}{3}e^{-(x+1)^2-y^2}$$

上述函数是 MATLAB 中函数 peaks 的定义，这是一个比较复杂的二元函数。

如图 10.2 所示，画出该二元函数在 z=0 上下两面的曲面图，从其中大致可以看出峰顶和谷底位置分布情况。

二元函数的极值计算过程是：

step 1 计算 $f(x, y)$ 的 5 个偏导数：

$$f_x(x, y) = \frac{\partial f(x, y)}{\partial x}, \quad f_y(x, y) = \frac{\partial f(x, y)}{\partial y}, \quad f_{xx}(x, y) = \frac{\partial^2 f(x, y)}{\partial x^2}, \quad f_{xy}(x, y) = \frac{\partial^2 f(x, y)}{\partial x \partial y},$$

$$f_{yy}(x, y) = \frac{\partial^2 f(x, y)}{\partial y^2}$$

step 2 求出方程组 $\begin{cases} f_x(x, y) = 0 \\ f_y(x, y) = 0 \end{cases}$ 的全部根，继而得到相应的驻点坐标。

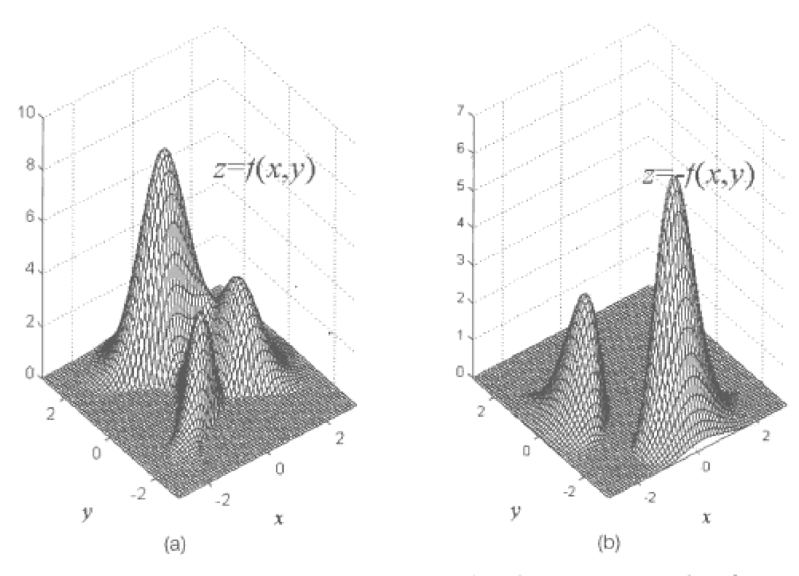


图 10.2　二元函数的图像：(a) $z=f(x,y)$，(b) $z=-f(x,y)$

step 3 分离出极大值和极小值。对于各个驻点坐标(x_k, y_k)，计算 3 个参数的值$A_k=f_{xx}(x_k, y_k)$，$B_k=f_{xy}(x_k, y_k)$，$C_k=f_{yy}(x_k, y_k)$，检验每个驻点判别式（$\Delta_k=A_kC_k-B_k^2$）的取值。当$\Delta_k>0$时该驻点是极值点：$A_k>0$对应着极小值，$A_k<0$对应着极大值；在$\Delta_k=0$时要进一步判断该点是否为极值点；当$\Delta_k<0$时，该点不是极值点。

函数 diff 和 jacobian 都可以用来计算二元函数的导数，这里使用 jacobian 函数计算导数。

```
syms x y;
z =  3*(1-x).^2.*exp(-(x.^2) - (y+1).^2) ...
   - 10*(x/5 - x.^3 - y.^5).*exp(-x.^2-y.^2) ...
   - 1/3*exp(-(x+1).^2 - y.^2);
dF=jacobian(z,[x,y]);
```

其中 dF 是一个 1×2 的符号型变量。为了求解 dF=0 这个方程组，需要使用 fsolve 函数计算，为此需要把 dF 转为字符串型变量。

```
S1=char(dF(1));          % 分离 dF 为 2 个字符串
S2=char(dF(2));          % 分离 dF 为 2 个字符串
S1=strrep(S1,'*','.*');      % 点运算转化
S1=strrep(S1,'^','.^');      % 点运算转化
S2=strrep(S2,'*','.*');      % 点运算转化
S2=strrep(S2,'^','.^');      % 点运算转化
fun=['[',S1,',',S2,']'];       % 字符串连接
fun=strrep(fun,'exp','q');    % 用字母 'q' 替代字符串 'exp'
fun=strrep(fun,'x','x(1)');    % 用 'x(1)' 代替 'x'
fun=strrep(fun,'y','x(2)');    % 用 'x(2)' 代替 'y'
fun=strrep(fun,'q','exp');    % 把 'exp' 替换回来
```

上面的程序完成了符号型数据向字符串的转化。其中为了避免字符串‘exp’中的‘x’被‘x(1)’替换，使用一个字母‘q’进行中转。

这里使用 ginput 函数用人机交互的方式求解极值位置。用 ginput 函数获得极值附近一个近似的点，作为 fsolve 函数输入的初始值。再调用 fsolve 函数进行精确的求解，同时所得的极值点与附件的点进行比较而分离出极大值和极小值点。相关实现程序如下：

```
contour(X,Y,Z,[-8:10]); % 绘制等高线
hold on;
set(gcf,'color','w')
set(gca,'Fontsize',12);
title('请点击等高线峰和谷的大致位置','Fontsize',12);
[xt,yt]=ginput(6)       % 鼠标取点
xt=xt';                 % 变为行向量
yt=yt';                 % 变为行向量
dx=[max(xlim)-min(xlim)]/20; % 产生一个 x 方向侧移
dy=dx;                  % 产生一个 y 方向侧移
for k=1:length(xt);
    x0=fsolve(fun,[xt(k),yt(k)],options);          % 求解方程组
    xt(k)=x0(1);                                    % 把当前精确根赋值到近似根 xt
    yt(k)=x0(2);                                    % 把当前精确根赋值到近似根 yt
    if subs(z,{x,y},{xt(k),yt(k)})>subs(z,{x,y},{xt(k)+dx,yt(k)+dy});% 函数值与附
近的点进行比较
        pb(k)=1;                                    % 标识极大值
    else
        pb(k)=-1;                                   % 标识极小值
    end
end
plot(xt(pb>0),yt(pb>0),'k^','MarkerFaceColor','k');  % 绘出极大值点
plot(xt(pb<0),yt(pb<0),'kv','MarkerFaceColor','k');  % 绘出极小值点
```

运行上述程序后在等高线上会出现一个“十”字形，使用鼠标在等高线上对应的峰值和谷底大致位置处单击即可获取当前点的坐标。依次单击后，程序将按所得各点的顺序逐个计算相应的极值点。上述程序计算得到的 xt，yt 和 pb 如下：

```
xt =   -0.0093   -1.3474    0.2964    1.2857   -0.4600    0.2283
yt =    1.5814    0.2045    0.3202   -0.0048   -0.6292   -1.6255
pb =     1    -1    -1     1     1    -1
```

输出如图 10.3 所示的图形。可以发现极值点已准确地计算出来，极大值和极小值分别用符号“▲”和“▼”标注出来。本例完整程序的文件名是 min_maxs2.m。

10.1.2 离散情况

对于离散情况，可以利用差分代替微分来求解极值，MATLAB 中的 diff 函数还可以用来计算差分。此外还可以通过相邻 3 个点的大小关系计算，如果中间一点同时大于（或者同时小于）两侧的点，那么这一点就是极大值（或者极小值）。

下面介绍两种方法计算向量型离散数据的极值。

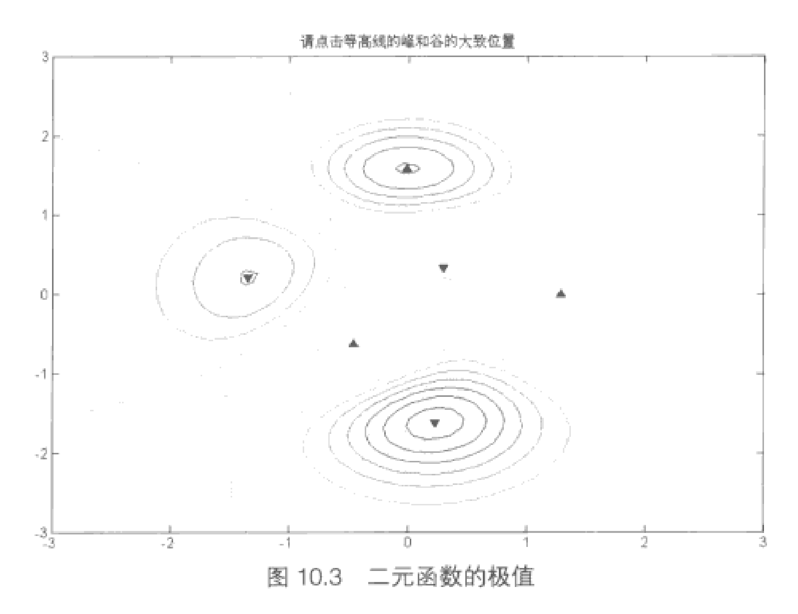


图 10.3　二元函数的极值

10.1.2.1　差分法

差分法，利用函数 diff 可以计算出差分而得到一个向量，如果该向量中相邻元素的符号相反，那么对应点就是一个极值点。其中前一个差分向量元素为正、后一个元素为负的情况对应着极大值，反之则为极小值。具体实现可以结合下面的例子理解。

```
A = [ 8, 2, 10, 7, 4, 3, 6, 9, 5, 1];       % 离散数据
pM = find(diff(sign(diff(A))) == -2)+1;  % 算出极大值的位置
pm = find(diff(sign(diff(A))) == 2)+1;   % 算出极小值的位置
AM = A(pM)                        % 返回极大值
Am = A(pm)                        % 返回极小值
```

上述程序运行结果如下：

```
AM =    10     9
Am =     2     3
```

极大值和极小值位置的求法还可以扩展为下面两种等价格式以及其他等价形式。

```
pM = intersect(find(diff(A)>0)+1,find(diff(A)<0));
pm = intersect(find(diff(A)<0)+1,find(diff(A)>0));
dA = diff(A);
dt = dA(1:end-1).*dA(2:end);
pM = find(dt<0 & dA(1:end-1)>0)+1;
pm = find(dt<0 & dA(2:end)>0)+1;
```

10.1.2.2　直接比较法

直接比较法，可以利用极大值（极小值）的定义来计算，如果相邻的 3 个数中，中间的数同时大于（小于）两边的数就是极大值（极小值）。

```
A = [ 8, 2, 10, 7, 4, 3, 6, 9, 5, 1];  % 离散数据
AL = A(1:end-2);
AM = A(2:end-1);
```

```
AR = A(3:end);
qM = find(AM>AL&AM>AR)+1;      % 找出中间同时大于两侧的位置
qm = find(AM<AL&AM<AR)+1;      % 找出中间同时小于两侧的位置
AM = A(qM)                     % 返回极大值
Am = A(qm)                     % 返回极小值
```

上述两种方法都支持向量计算没有循环，执行速度很快，完全可以用于大量数据计算。读者可以把它们写为函数的形式，在自己的计算中调用。此外，还可以用求最大值的方法求最小值，即计算-A 的最大值位置就是计算 A 最小值的位置，反之亦然。

而对于矩阵形式的数据，可以根据极大值和极小值的关系来寻找位置和极值。如图 10.4 所示，在一个3×3 的局域内，描述了出现极大值和极小值的情况。

m_1	m_2	m_3
m_4	M	m_5
m_6	m_7	m_8

M_1	M_2	M_3
M_4	m	M_5
M_6	M_7	M_8

图 10.4 二维离散数据极大值和极小值的关系

在图 10.4 所示的局域子矩阵中，如果$M > m_k$，$(k=1,2,...,8)$都成立，那么数值M 就表示一个极大值；同样地，如果$m < M_k$，$(k=1,2,...,8)$均成立，那么数值m 就是一个极小值。根据这个关系可以编程逐个计算出8×8 子矩阵的极大值。

```
rand('state',12);                    % 设置随机数状态
A=reshape(randperm(64),8,8);          % 产生一个 8×8 矩阵
A1 = A(1:end-2,1:end-2);              % 分离出左上方子矩阵
A2 = A(1:end-2,2:end-1);              % 分离出正上方子矩阵
A3 = A(1:end-2,3:end-0);              % 分离出右上方子矩阵
A4 = A(2:end-1,1:end-2);              % 分离出左中方子矩阵
A5 = A(2:end-1,3:end-0);              % 分离出右中方子矩阵
A6 = A(3:end-0,1:end-2);              % 分离出左下方子矩阵
A7 = A(3:end-0,2:end-1);              % 分离出中下方子矩阵
A8 = A(3:end-0,3:end-0);              % 分离出右下方子矩阵
Am = A(2:end-1,2:end-1);              % 分离出中间的子矩阵
[x,y] = find(Am>A1&Am>A2&Am>A3&Am>A4&...
   Am>A5&Am>A6&Am>A7&Am>A8);          % 判断大小关系并得到极大值
P = [x'+1;y'+1];                     % 得到极大值的位置坐标
ind = sub2ind(size(A),x+1,y+1);       % 把坐标转化为索引
AMm = A(ind)';                       % 得到极大（小）值
A, P, AMm                           % 显示计算结果
```

结果输出为：

```
A =    14    52    11    55    10     7    58    23
       19    64    47    12    30    13    48    20
       25    31    35    43    29    59    63     6
       21    33    17    42     9    18    16    26
       28    38    50    34    62    24    61    57
       37    54    60     1    45    51    36     4
        5    49    40    15'   46     2    22    56
```

```
      53    44    39    27     3    41    32     8
P =     2     6     5     3     5
        2     3     5     7     7
AMm =  64    60    62    63    61
```

对比上面的矩阵，上述这段程序准确地计算出了相应极大值的位置以及极大值。为了给出一个更直观的观察，图 10.5 显示了矩阵 A 灰度图，其中白色位置处的数值大，而黑色位置处数值小。根据旁边的刻度我们也可以读出极大值的位置。如果把含 find 的语句替换为下面的形式：

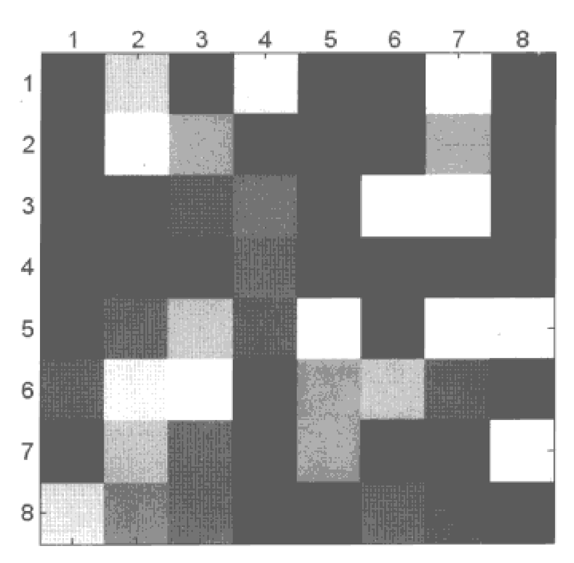


图 10.5 矩阵 A 的图像显示

```
[x,y] = find(Am<A1&Am<A2&Am<A3&Am<A4&...
    Am<A5&Am<A6&Am<A7&Am<A8);
```

输出的极小值信息是：

```
P =     4     6     4     7
        3     4     5     6
AMm =  17     1     9     2
```

上述程序也可以对任意矩阵进行求极值计算，读者稍加修改就可以用来计算大型矩阵的极值。本问题的完整求解程序对应的文件名是 min_maxs2D.m。相比之下使用差分形式的程序书写要复杂得多，读者如感兴趣可以考虑这个问题。

10.2 最值

前面所述的极值问题得到的是某区间上的所有极值，求出最小值是获得某定义范围内的一个最小值大小以及相应位置。极值可以看做小局域内的最大值或者最小值。求 $f(x)$ 最大值问题可以通过计算 $-f(x)$ 的最小值问题而得到解决，这里 x 可以是包含多个参数的向量。

10.2.1 离散数据的最值

在 MATLAB 中提供了两个基本的函数 max 和 min 来计算离散数据的最大值和最小值，它们的调用格式如下：

```
Y = max (X)          % Y 是返回的最大值，X 是向量
[Y, I] = max (X)       % Y 是返回的最大值，I 是最大值对应的位置，X 是向量
Z = max(X, Y)        % Z 是选择矩阵 X 和 Y 各个位置处的最大值组成的矩阵
[Y, I] = max(X, [], 1)  % 计算矩阵 X 各列的最大值：Y 是最大值，I 是各列中最大值对应的行位置
[Y, I] = max(X, [], 2)  % 计算矩阵 X 各行的最大值：Y 是最大值，I 是各行中最大值对应的列位置
Y = min (X)           % Y 是返回的最小值，X 是向量
[Y, I] = min(X)        % Y 是返回的最小值，I 是最小值对应的位置，X 是向量
Z = min(X, Y)         % Z 是选择矩阵 X 和 Y 各个位置处的最小值而组成的矩阵
[Y, I] = min(X, [], 1)   % 计算矩阵 X 各列的最小值：Y 是最小值，I 是各列中最小值对应的行位置
[Y, I] = min(X, [], 2)   % 计算矩阵 X 各行的最大值：Y 是最小值，I 是各行中最小值对应的列位置
```

下面以一个例子给出 X 中含多个最大值的时候 max 函数计算结果的说明。

```
X = [1, 2, 3, 3, 1, 2];        % 可以发现 3 是最大值
[Y, I] = max(X)              % 计算最大值
```

输出的结果为：

```
Y =     3
I =     3
```

可以发现只有第一个最大值被返回。而有的时候需要返回多个最大值，此时需要使用下面的语句：

```
I = find(X==max(X))     % 找出 X 中等于最大值的点
Y = X(I)               % 按各个位置提取出最大值
```

输出结果为：

```
I =     3     4
Y =     3     3
```

但是在有些情况下，函数表达式期望的数值并不严格地相等，即

```
3*tan(pi/4)
```

输出结果为：

```
ans =3.0000
```

如果用下面的方式来计算：

```
3*exp(i*pi*2)
```

输出结果为：

```
ans =   3.0000 - 0.0000i
```

上面两个表达式的结果都是 3，希望通过求最大值操作把这 3 个数都找到。在这样的情况下，

使用运算“= =”不能获得正确的结果，因此计算最大值的时候需要使用一个很小的数值来等效地替换“相等”计算，即

```
X = [1, 2, 3, 3*tan(pi/4), 3*exp(i*pi*2)];    % 含表达式的数组
I = find(abs(X)>max(X)-1e-6)          % 用不等号代替等于来判断大小关系
Y = X(I)                          % 按各位置提取出最大值
```

输出结果为：

```
I =     3     4     5
Y =   3.0000             3.0000             3.0000 - 0.0000i
```

关于 min 函数的使用与 max 类似，这里不再赘述，读者可以参考 MATLAB 帮助文档。

10.2.2　连续函数的最小值

如前所述，连续函数的最大值计算可以转化为极小值问题进行求解。本小节结合 MATLAB 最优化工具箱提供的专门函数介绍最小值的计算。相关函数如表 10.1 所示。

表 10.1　计算最小值的相关函数

函数名	说明	备注
optimset	设定优化过程参数	可用于很多函数
optimget	获取优化过程参数	
fminsearch	利用单纯形法求最小值	无约束条件
fminunc	利用拟牛顿法求最小值	
linprog	线性规划	有约束条件
fminbnd	一元非线性规划	
fmincon	多元非线性规划	
quadprog	二次规划	
fseminf	半无限多元规划	
fmininax	最大最小化模型	
fgoalattain	多目标规划	
lsqlin	最小二乘最优问题	

对于优化过程控制参数，MATLAB 提供了函数 optimset 和 optimget 来分别设置和查看相关的参数。具体参数意义如表 10.2 所示。

表 10.2　优化函数中相关参数的意义说明

参数名	说明	默认值
DerivativeCheck	将用户提供的导数和有限差分求出的导数进行对比	off
Diagnostics	打印将要最小化或求解的函数的诊断信息	off
DiffMaxChange	变量中有限差分梯度的最大变化	1e-1
DiffMinChange	变量中有限差分梯度的最小变化	1e-8
Display	显示水平	off
GoalsExactAchieve	使得目标个数刚好达到，不多也不少	0
GradConstr	用户定义的约束函数的梯度	off
GradObj	用户定义的目标函数的梯度。使用大型方法时必须使用梯度，对于中型方法则是可选项	off

（续表）

参数名	说明	默认值
Hessian	用户定义的目标函数的 Hessian 矩阵	off
HessMult	用户定义的 Hessian 乘法函数	[]
HessPattern	用于有限差分的 Hessian 矩阵的稀疏形式	sparse matrix
HessUpdate	拟牛顿法	bfgs
Jacobian	用于用户定义的目标函数的 Jacobian 矩阵	off
JacobMult	用户定义的 Jacobian 乘法函数	[ ]
JacobPattern	用于有限差分的 Jacobian 矩阵的稀疏形式	sparse matrix
LargeScale	选用大型算法	on
LevenbergMarquardt	用 Levenberg-Marquardt 法替代 Gauss-Newton 法	off
LineSearchType	选择插值方法	quadcubic
MaxFunEvals	函数评价的允许最大次数	[ ]
MaxIter	函数迭代的允许最大次数	**
MaxPCGIter	PCG 迭代的最大次数	**
MeritFunction	使用目标达到或最大最小化目标函数的方法	multiobj
MinAbsMax	F(x)最坏绝对值最小化了的目标数	0
PrecondBandWidth	PCG 前处理的上带宽	0
TolCon	约束矛盾的终止容限	**
TolFun	函数值处的终止容限	**
TolPCG	PCG 迭代的终止容限	0.1
TolX	X 处的终止容限	**
TypicalX	典型 X 值	**

例 10–1：计算函数 $f(x,y)=-\left[(x-1)^2+(y+1.2)^2\right]e^{-\frac{x^2+y^2}{4}}$ 在(-1, 1)附近的最小值。

为了便于比较，在计算最小值之前，需要先用 meshc 函数把该二元函数对应的曲面画出来。

```
[X, Y] = meshgrid(linspace(-3,3,20));
Z = -[(X-1).^2+[Y+1.2].^2].*exp(-[Y.^2+X.^2]/4);
```

所得的图形如图 10-6 所示。

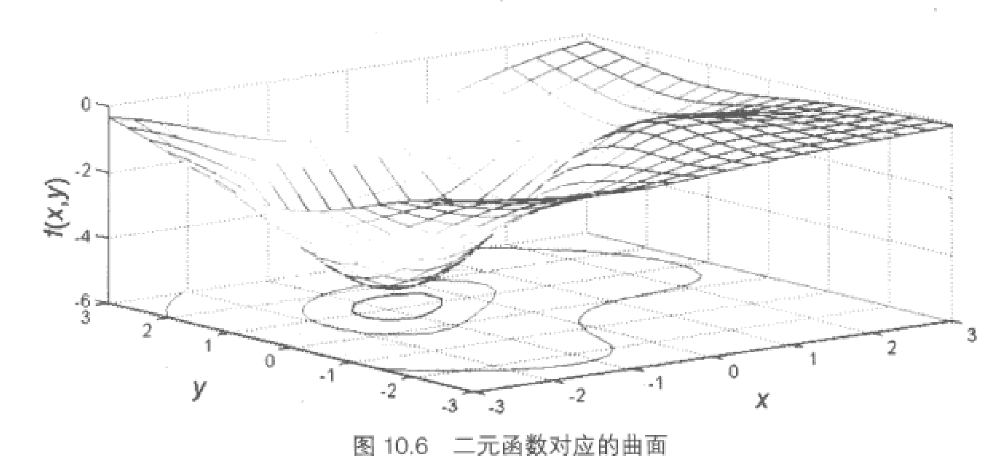


图 10.6　二元函数对应的曲面

利用下面的两条语句可以实现单纯形法计算最小值：

```
fun=inline('-[(x(1)-1)^2+(x(2)+1.2)^2]*exp(-[x(1)^2+x(2)^2]/4)');     % 定义二元
函数
[xm, Fval]=fminsearch(fun,[-1;1])                                 % 计算最小值位置和最小值
```

程序输出：

```
xm =   -0.8745
        1.0495
Fval =   -5.3773
```

这里二元函数 $f(x, y)$ 中的变量 (x, y) 要分别用“x(1)”和“x(2)”这样的方式输入，不能以“x”和“y”形式书写在 inline 函数里面。否则程序将提示下面的错误：

```
??? Error using ==> inline/feval
Not enough inputs to inline function.
```

使用函数 fminunc 来解决上面的例 10-1，所用的语句如下：

```
[xm,Fval] = fminunc(fun,[-1;1])
```

执行上面的语句有下面的输出：

```
xm =   -0.8750
        1.0495
Fval =   -5.3773
```

可以发现这两个函数计算的结果是相同的。在光盘中解决上述问题的计算完整程序名是 fminsearch_test.m。



如果函数中的变量幂次大于 2，则使用 fminunc 要比 fminsearch 有效，但是当所选函数高度不连续或者变化剧烈时，使用 fminsearch 较好。

10.2.2.1 线性规划问题

线性规划问题的求解方法有表上作业法、图解法和单纯形法，然而在决策变量比较多的时候，上述方法的求解过程都是比较复杂的，使用 MATLAB 求解线性规划问题却比较简单。

使用 MATLAB 求解线性规划问题的函数是 linprog，函数名源自英文 Linear programming。其调用格式有下面 3 种样式：

```
X = linprog(f, A, b)
[X, Fval] = linprog(f, A, b, Aeq, Beq, Lb, Ub)
[X, Fval, exitflag, output, lambda] = linprog(f, A, b, Aeq, Beq, Lb, Ub, X0, options)
```

输出参数说明：这里 X 是由决策变量组成的列向量。Fval 是优化结束后得到的目标函数的数值大小。返回值 exitflag 具有 3 种可能的情况，即 0 表示优化结果已经超过了函数的估计值或者已声明的最大迭代次数，1 表示优化过程中变量收敛到最终结果 X，-1 表明优化结果不收敛。返回值 output 具有 3 个分量，即 iterations 是优化过程的迭代次数，cgiterations 表示 PCG 迭代次数，algorithm 是优化采用的运算规则。返回值 lambda 有 4 个分量，即 ineqlin 表示线性不等式约束条

件，eqlin 是线性等式约束条件，upper 是变量的上界约束条件，lower 是变量的下界约束条件。它们的返回值分别表示相应的约束条件在最优化过程中是否有效。

输入参数说明： f 是由目标函数中变量系数依次组成的向量。A 是一个矩阵，b 是一个向量，它们组成线性规划问题的不等式约束条件为 $AX \le b$。Aeq 和 Beq 组成线性规划问题的等式约束条件为 $A_{eq}X = B_{eq}$。Lb 和 Ub 分别是变量的下界和上界约束。X0 是变量的初值。options 是控制线性规划过程的系列参数。

调用函数 linprog 时，系统默认该函数的所有输入 f，A，b，Aeq，Beq，Lb，Ub，X0，options 都是存在的，且按描述顺序进行赋值。比如当 Lb 存在的情况下，它后面的参数可以不写入，程序不会出错。但是 Lb 前面的参数必须给出，即使问题中没有相关参数，也要使用空矩阵“[ ]”进行赋值，不能缺省。如果线性规划问题不存在可行解时，将会出现如下提示：

```
Warning: The constraints are overly stringent;there is no feasible solution.
```

如果存在收敛解，会显示这样的文字：

```
Optimization terminated successfully.
```

例 10-2： 求解下面的线性规划问题。

min f：$3x_1 + 2x_2$

约束条件为：
$$\begin{cases} 2x_1 - x_2 \ge 1 \\ x_1 < 4 \\ x_2 < 3 \\ x_1, x_2 \ge 0 \end{cases}$$

分析： 在这个问题中，f=[3,2]，约束不等式参数 A=[-2,1]，b= -6，下界 Lb=[0,0]，上界 Ub=[4,3]。现在可以根据这些条件编写优化程序。相应程序如下：

```
f = [3, 2];                  % 定义目标函数
A = [-2, 1];                 % 约束不等式中变量前面的系数
b = -1;                 % 约束不等式中 b 的值
Lb = [0, 0];                 % 变量的下界
Ub = [4, 3];                 % 变量的上界
[X, Fval] = linprog(f, A, b,[],[], Lb, Ub)    % 执行线性规划计算
```

上述程序输出下面的内容：

```
X =    0.5000
      0.0000
Fval =1.5000
```

例 10-3： 求解下面的线性规划问题。

min f：$x_1 + 2x_2 + x_3$

约束条件为：$\begin{cases}5x_1+3x_2+2x_3=3\\2x_1+3x_2+4x_3=5\\x_1,x_2,x_3\geq 0\end{cases}$

分析：这里 f=[1,2,1]，约束方程的参数 Aeq=[5,3,2；2,3,4]，Beq=[3; 5]，下界 Lb=[0,0,0]，其他约束类型取默认值即可。求解时采用[X, Fval] = linprog(f, A, b, Aeq, Beq, Lb, Ub)形式。程序如下：

```
f = [1, 2, 1];                   % 定义目标函数
Aeq = [5, 3, 2; 2, 3, 4];        % 约束方程中变量前面的系数
Beq = [3; 5];                    % 约束方程的系数
Lb = [0, 0, 0];                  % 变量的下界
[X, Fval] = linprog(f, [], [], Aeq, Beq, Lb)     % 执行线性规划计算
```

输出如下：

```
X =    0.1250
       0.0000
       1.1875
Fval =    1.3125
```

例 10-4：求解下面的线性规划问题。

min f：$2x_1+3x_2+5x_3+6x_4$

约束条件为：$\begin{cases}2x_1-x_2+2x_3-x_4\geq 20\\3x_1+2x_2+3x_3\leq 12\\2x_1+3x_2+x_3+2x_4=15\\x_1,x_2,x_3,x_4\geq 0\end{cases}$

分析：这里 f=[2, 3, 5, 6]，约束不等式的参数 A=[-2, 1, -2, 1; 3, 2, 1, 0]，b=[-20； 12]，约束方程的参数 Aeq=[2,3,1,2]，Beq=15，这里仅有下界描述 Lb=[0,0,0,0]。求解时采用[X, Fval] = linprog(f, A, b, Aeq, Beq, Lb, Ub)形式。程序如下：

```
f = [2, 3, 5, 6];                              % 定义目标函数
A = [-2, 1, -2, 1; 3, 2, 1, 0];                     % 约束不等式中变量前面的系数
b = [-20, 12];                                 % 约束不等式中 b 的值
Aeq = [2, 3, 1, 2];                            % 约束方程中变量前面的系数
Beq = 15;                                      % 约束方程的系数
Lb = [0, 0, 0, 0];                             % 变量的下界
[X, Fval] = linprog(f, A, b, Aeq, Beq, Lb)    % 执行线性规划计算
```

输出结果如下：

```
X =    0.4716
       0.0840
       10.4173
       1.6938
Fval =  63.4444
```

在上述例 10-4 中，如果把约束不等式 $2x_1 - x_2 + 2x_3 - x_4 \geq 20$ 变为 $2x_1 - x_2 + 2x_3 - x_4 \geq 40$，其他条件不变，那么进行线性规划时将会出现下面的输出：

```
X =    2.7859
      0.0001
      12.4485
      0.0001
Fval =   67.8150
```

在这种情况下，只是换了约束不等式，其他条件未变，说明约束条件存在问题，从而导致规划失败。

10.2.2.2 非线性规划问题

前面介绍了线性规划的求解，本节介绍非线性规划问题的求解方法。根据目标函数和约束条件的不同，MATLAB 提供了 fminbnd，fmincon，quadprog，fseminf，fminimax，fgoalattain 及 lsqlin 等函数来求解不同类型的非线性规划问题。下面结合实例来介绍这些函数的用法。

在 MATLAB 中可以用函数 fminbnd 来求解一元非线性规划问题，其调用格式为：

```
x = fminbnd(fun, x1, x2);
[x, fval] = fminbnd(fun, x1, x2);
[x, fval, exitflag, output] = fminbnd(fun, x1, x2, options);
```

参数说明： x 是最小值对应的变量数值，fval 是目标函数的最小值。exitflag 是终止迭代条件，即 exitflag=1 表示函数收敛于 x，exitflag=0 表示超过函数估计值或迭代的最大数字，exitflag=-1 表示函数不收敛于 x。output 为优化输出信息，它包含 3 项信息，即 iterations 为迭代次数，funccount 为函数赋值次数，algorithm 为使用的算法。fun 是目标函数的字符串或者内联函数 inline 的句柄，此外对于非常复杂的目标函数表达式可以书写函数文件来定义。x1 和 x2 分别为变量的下界和上界。options 是优化参数选项。

例 10-5： 求下面函数在区间(1, 5)内的最小值。

$$f(x) = \frac{\sin x}{x^2} + 3x\cos x$$

分析： 首先需要定义需要求解的函数 fun，然后用[x, fval] = fminbnd(fun, x1, x2)来进行求解。

根据上面介绍函数 fminbnd 的用法，编写如下程序：

```
fun = inline('[sin(x)/x^2+3*x*cos(x)]');      % 定义函数，也可以写为 "fun
='[sin(x)/x^2+3*x*cos(x)]';"
[x, fval, exitflag, output] = fminbnd(fun, 1, 5)  % 进行非线性规划
```

程序输出下面结果：

```
x =    3.4314
fval =   -9.8892
exitflag =     1
output =     iterations: 9
          funcCount: 11
```

函数 fmincon 可以用来求解多元非线性规划问题，其调用格式为：

```
[x, fval, exitflag, output, lambda, grad, hessian] = fmincon(fun, x0, A, b, Aeq,
Beq, Lb, Ub, nonlcon, options);
```

参数说明：参数 x，fval，exitflag 意义同前面的函数 fminbnd。output 给出了输出信息，包括迭代次数 iterations、函数赋值次数 funccount、步长 stepsize、所使用的算法 algorithm、一阶优化的度量 firstorderopt、PCG 迭代次数 cgiterations 等。lambda 是 Lagrange 乘子，它决定哪一个约束有效：下界约束 lower、上界约束 upper、线性等式约束 eqlin、非线性等式约束 eqnonlin、线性不等式约束 ineqlin、非线性不等式 ineqnonlin。grad 是目标函数在 x 处的梯度。hessian 是目标函数在 x 处的 Hessiab 值。A，b 和 Aeq，Beq 是决定参数范围约束不等式和等式的系数矩阵。Lb 和 Ub 分别是下界和上界。nonlcon 通过接受的向量 x 来计算非线性不等约束 $C(x)\le 0$ 和等式约束 $C_{eq}(x)=0$ 分别在 x 处的估计 C 和 Ceq，通过指定函数柄来使用（其中非线性不等式和等式约束在 nonlcon.m 中定义，具体使用见下面的例子）。options 参数可以通过 optimset 函数来设定。

例 10–6：求解下面的非线性函数。

$$f(x,y)=\sin\left(x^2-3y\right)+e^{-x^2}\cos y$$

约束条件为：$\begin{cases}-x+(y-2)^2\ge 0\\ x-2y+1\ge 0\end{cases}$

分析：在求解时，可以先定义约束不等式，然后定义求解函数 fun，最后用多元非线性规划函数来求解。本问题中约束条件只有非线性和线性不等式，其他约束条件可以选择为空。

其中非线性不等式约束要通过下面的文件定义：

```
function [C,Ceq]=nonlcon1(x);
C=x(1)-[x(2)-2].^2;        % 非线性不等式约束
Ceq=[];                    % 把非线性等式设置为空
```

主程序可如下书写：

```
fun = 'sin(x(1)^2-3*x(2))+exp(-x(1)^2)*cos(x(2))';
x0 = [1.2, 1];
A = [-1, 2];
b = 1;
[x, fval, exitflag, output, lambda, grad, hessian] =...
   fmincon(fun, x0, A, b, [], [], [], [], @nonlcon1)
```

上述程序的执行结果是：

```
Active Constraints:     2
x =     1.1398    0.9324
fval =    -0.8348
exitflag =     1
output =     iterations: 4
          funcCount: 20
          stepsize: 1
 algorithm: 'medium-scale: SQP, Quasi-Newton, line-search'
          firstorderopt: 2.6799e-005
          cgiterations: []
lambda =    lower: [2x1 double]
          upper: [2x1 double]
          eqlin: [0x1 double]
```

```
          eqnonlin: [0x1 double]
          ineqlin: 0
          ineqnonlin: 0.2047
grad =   -0.2048
-0.4372
hessian =
    5.9278   -6.2136
   -6.2136    8.6473
```

例 10-7：求下面函数在初始点 x0=（0，1，1）处的最小值。

$$f(x,y,z)=xy+yz+zx$$

约束条件为：$\begin{cases}2x+y+2z=2\\x^2+y^2+z^2=4\end{cases}$

分析：对于本问题的求解，需要先定义非线性约束等式，其由一个专门的函数文件来完成，然后定义目标函数 fun，最后用多元非线性规划函数来求解。这里约束条件是等式型，其他不等式约束条件可以设置为空。

其中非线性等式约束要通过下面的文件定义：

```
function [C,Ceq]=nonlcon2(x);
C = [];                          % 非线性不等式约束
Ceq = x(1)^2+x(2)^2+x(3)^2-4;  % 把非线性等式设置为空
```

相关主程序如下：

```
fun = 'x(1)*x(2)+x(2)*x(3)+x(1)*x(3)';
x0 = [0, 1, 1];
Aeq = [2, 1, 2];
Beq = 2;
[x, fval] = fmincon(fun, x0, [], [], Aeq, Beq, [], [], @nonlcon2)
```

上述程序输出：

```
Active Constraints:
     1
     2
x =    0.8889   -1.5556    0.8889
fval =   -1.9753
```

这表明一个收敛的最小值被求出。

函数 quadprog 可以用来求解二次规划问题，其完整调用格式为：

```
[x, fval, exitflag, output, lambda] = quadprog(H, f, A, b, Aeq, Beq, Lb, Ub, x0,
options);
```

参数说明：参数 x，fval，exitflag，output，lambda，A，b，Aeq，Beq，Lb，Ub，x0，options 与函数 fmincon 和 fminbnd 的用法相同。参数 H 和 f 是二次型中的参数，具体定义可以从下面的例子获得。

例 10-8：求解下面的二次规划问题。

$$f(x_1,x_2)=2x_1^2+x_2^2+3x_1x_2-2x_1-5x_2$$

约束条件为：$\begin{cases} 2x_1+5x_2 \le 6 \\ 7x_1-2x_2 \ge -1 \\ x_1+3x_2 \le 7 \\ x_1 \ge 0, x_2 \ge 1 \end{cases}$

分析：对于本问题的求解，约束条件含有线性约束不等式和下界条件，目标函数是一个二次型函数。在定义了目标函数和约束条件之后，可以调用多元非线性规划函数来求解。而二次型的标准格式为 $f(x)=\frac{1}{2}x'Hx+f'x$，其中 $x=[x_1,x_2]$，令 $H=\begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix}$，$f=\begin{bmatrix} f_1 \\ f_2 \end{bmatrix}$，有 $f(x)=\frac{1}{2}[x_1 \quad x_2]\begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}+[x_1 \quad x_2]\begin{bmatrix} f_1 \\ f_2 \end{bmatrix}=\frac{1}{2}\left[h_{11}x_1^2+\left(h_{12}+h_{21}\right)x_1x_2+h_{22}x_2^2\right]+f_1x_1+f_2x_2$。其中取 $h_{12}=h_{21}$，我们对比问题中的二次型可以得出矩阵 $\boldsymbol{H}$ 和向量 $\boldsymbol{f}$ 的取值，$H=\begin{bmatrix} 2 & 3 \\ 3 & 4 \end{bmatrix}$，$f=\begin{bmatrix} -2 \\ 4 \end{bmatrix}$。约束条件的系数类似前面的例子即可确定出来。程序如下：

```
H = [4, 3; 3, 2];
f = [-2, -5];
A = [2, 5; -7, 2; 1, 3];
b = [6; 1; 7];
Lb = [0, 1];
[x, fval, exitflag, output, lambda] = quadprog(H, f, A, b, [], [], Lb)
```

这段程序输出结果是：

```
x =    0.1795
      1.1282
fval =   -4.0552
exitflag = 1
output =  iterations: 2
 algorithm: 'medium-scale: active-set'
         firstorderopt: []
         cgiterations: []
lambda =  lower: [2x1 double]
         upper: [2x1 double]
         eqlin: [0x1 double]
ineqlin: [3x1 double]
```

通过检查 lambda 下面 4 项可以知道相关约束是否有效：

```
lambda.lower
ans =     0
         0
lambda.upper
ans =     0
         0
lambda.ineqlin
ans =    0.2880
        0.3826
        0
```

0 表示相应约束无效，因此上面结果表明仅线性约束的前两项有效，其他约束无效。

为了方便确定二次型系数矩阵，作者设计了函数文件 quadcoeff.m，该函数的调用方法如下：

```
fun = 'x^2+y^2+3*x*y-2*x-5*y';
[H1, f1] = quadcoeff(fun)
fun = 'x^2+y^2+2*z^2+3*x*y-2*x-5*y+z';
[H2, f2] = quadcoeff(fun)
```

返回下面的结果：

```
H1 =     2     3
         3     2
f1 =     -2    -5
H2 =     2     3     0
         3     2     0
         0     0     4
f2 =    -2    -5     1
```

先是对函数表达式各变量求一次导数，再把所有变量赋值为 0，就可以得到 f 对应的系数；依次对各变量求二次偏导数将得到 H 的系数。在使用函数 quadcoeff 的时候应该注意：变量的顺序是按英文字母的顺序排列的，即 a~z，而且函数表达式里面的变量只能用小写字母，此外函数表达式需要合法。利用这个函数就可以方便地计算出二次型问题中的系数矩阵了。

例 10-9：计算下面的二次规划问题。

$$f\left(x_1,x_2,x_3,x_4\right)x = x_1^2+2x_2^2+3x_3^2+4x_4^2+x_1x_2+x_3x_4-x_2+x_3$$

约束条件为：$\begin{cases} x_1+x_4=1 \\ 2x_2+x_3=2 \end{cases}$

分析：对于本问题的求解，约束条件只是线性约束等式，目标函数是一个二次型函数。在定义了目标函数和约束条件之后，可以调用多元非线性规划函数来求解。而二次型系数矩阵的提取可以调用前面定义的函数 quadcoeff 来计算。程序如下：

```
fun = 'u^2+2*x^2+3*y^2+4*z^2+x*y+y*z-u+y';  % 对应关系是 u→x1，x→x2，y→x3，
z→x4
[H,f]=quadcoeff(fun);                          % 计算二次型系数
[x, fval] = quadprog(H, f, [], [], [1, 0, 0, 1;0, 2, 1, 0], [1,2])  % 二次规划
```

上述程序输出如下：

```
x =    0.9175
       0.9924
       0.0152
       0.0825
fval =1.9535
```

函数 fseminf 可以用来求解半无限有约束多元函数最优解问题，其调用格式为：

```
[x,fval,exitflag, output,lambda] =
fseminf(fun,x0,Ntheta,seminfcon,A,b,Aeq,beq,lb,ub,options)
```

参数说明：参数 x，fval，exitflag，output，lambda，fun，x0，A，b，Aeq，beq，lb，ub，options 的意义同前面的函数，这里不再赘述。Ntheta 是半无限约束的个数。seminfcon 用来确定非线性不等式和等式约束函数文件名，具体定义可参考下面的例子。

例 10-10：求下面的最优化问题。

$$f(x_1,x_2,x_3)=(x_1-0.2)^2+(x_2-0.3)^4+(x_3-0.4)^6$$

约束条件为：$\begin{cases}\Phi_1(x_1,x_2,x_3,w_1)=\sin(w_1x_1)-0.1+2\cos(w_1x_2)+4\sin(w_1x_3)\le 2\\ \Phi_2(x_1,x_2,x_3,w_2)=\sin(w_2x_1)-0.2+2\sin(w_2x_2)+3\cos(w_2x_3)\le 3\end{cases}$

其中 w_1 是奇数，且 $1\le w_1\le 9$。w_2 是偶数，且 $2\le w_2\le 8$。

分析：本问题的约束条件只是半无限约束，这个约束条件需要专门定义一个函数来实现。目标函数是一个非线性函数。在定义了目标函数和约束条件之后，可以调用多元非线性规划函数 fseminf 来求解。首先编写函数文件 myseminfcon.m 定义这两个半无限约束条件。

```
function [C,Ceq,PHI1,PHI2,S] = myseminfcon(X,S);
if  isnan(S(1,1)); % 定义半无限有约束
   S = [2, 0; 2, 0];  % 初始化样本间距
end
w1 = 1:S(1,1):9;    % 产生样本集
w2 = 2:S(2,1):8;
PHI1 = sin(w1*X(1))- % 计算半无限约束
0.1+2*cos(w1*X(2))+4*sin(w1*X(3))-2;
PHI2 = sin(w2*X(1))-0.2+2*sin(w2*X(2))+4*cos(w2*X(3))-3;
C = [ ]; Ceq=[ ];                 % 设置非线性约束为空
plot(w1,PHI1,'-',w2,PHI2,':');      % 绘制半无限约束条件曲线
title('Semi-infinite constraints');
```

下面是主程序：

```
fun = '[x(1)-0.2]^2+[x(2)-0.3]^4+[x(3)-0.4]^6';        % 定义目标函数
x0 = [0.1,0.1,0.1];                                   % 设置初值
[x,fval] = fseminf(fun,x0,2,@myseminfcon)           % 进行优化计算
```

输出如下结果

```
Active Constraints:
     1
     2
x =    0.0331   -0.2221    0.0290
fval =  0.1048
```

输出半无限约束条件曲线如图 10.7 所示。

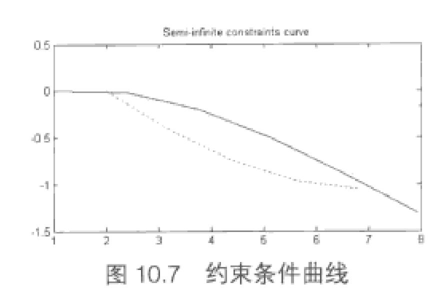


图 10.7　约束条件曲线

函数 fminimax 可以用来计算极小化极大（Minmax）问题，其调用格式如下：

```
[x,fval,maxfval,exitflag,output,lambda] =
fminimax(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon, options)
```



这里参数 fval 是目标函数的最小值，maxfval 是目标函数在 x 处的最大值，其他参数与前面介绍的相同。

例 10-11：求下面函数最大值的最小化问题。

$$F(x)=\left[f_1(x),\ f_2(x),\ f_3(x),\ f_4(x),\ f_5(x)\right],\ x=\left[x_1,x_2\right]$$

$$\text{其中}\begin{cases} f_1(x)=x_1^2+2x_2^2+4x_1-7x_2 \\ f_2(x)=x_1^2-4x_2^5 \\ f_3(x)=x_1+2x_2 \\ f_4(x)=x_1-2x_2 \\ f_5(x)=2x_1+5x_2 \end{cases}$$

分析：本问题的目标函数是由多个函数组成的，是一个最大值最小化问题。目标函数的定义需要建立一个函数文件。不存在约束条件。在定义了目标函数之后，就可以调用多元非线性规划函数 fminimax 来求解了。利用下面的程序定义目标函数：

```
function y = myfun3(x)
y(1)= x(1)^2+2*x(2)^2+4*x(1)-7*x(2);
y(2)= x(1)^2-4*x(2)^5;
y(3)= x(1)+2*x(2);
y(4)= x(1)-2*x(2);
y(5)= 2*x(1)+5*x(2);
```

主程序如下：

```
x0 = [0.1; 8];
[x,fval,maxfval] = fminimax(@myfun3,x0)
```

计算结果如下：

```
Active Constraints:
     1
     2
     3
x =    -4.8288
      1.4456
fval =   -1.9375   -1.9375   -1.9375   -7.7201   -2.4295
maxfval = -1.9375
```

函数 fgoalattain 可以用来计算多目标规划问题，其完整调用格式如下：

```
[x,fval,attainfactor,exitflag, output,lambda] =
fgoalattain(fun,x0,goal,weight,A,b,Aeq,beq,lb, ub,nonlcon,options);
```

参数说明：goal 是用户设计的目标函数值向量；weight 是权值系数向量，用于控制目标函数

与用户自定义目标值的接近程度；fval 是多目标函数在 x 处的值；attainfactor 为解 x 处的目标规划因子。其他参数与前面函数的意义相同。

例 10-12：计算一个线性微分方程系统，设计控制系统输出反馈器 $\begin{cases}\dot{x}=(A+BKD)x+Bu \\ y=Dx\end{cases}$，

其中 $A=\begin{bmatrix}-0.5 & 0 & 0 \\ 0 & -2 & 10 \\ 0 & 1 & -2\end{bmatrix}$，$B=\begin{bmatrix}1 & 0 \\ -2 & 2 \\ 0 & 1\end{bmatrix}$，$D=\begin{bmatrix}1 & 0 & 0 \\ 0 & 0 & 1\end{bmatrix}$。

要求该系统在复平面实轴上点[-5,-3,-1]的左侧有极点，同时 $|K_{i,j}|\le 4$，$(i,j=1,2)$。

分析：这个问题中矩阵 $\boldsymbol{K}$ 是未知的，通过设计“系统在复平面实轴上点[-5, -3, -1]的左侧有极点”这一条件来建立优化任务。这个问题属于多目标规划问题，约束条件为矩阵 $\boldsymbol{K}$ 各元素存在上下界。在定义了目标函数和约束条件之后就可以调用多目标规划函数 fgoalattain 来求解了。本问题相关程序如下：

```
fun = inline('sort(eig(A+B*K*D))','K','A','B','D');  % 定义目标函数
A = [-0.5 0 0; 0 -2 10; 0 1 -2];
B = [10; -2 2; 0 1];
D = [10 0; 0 0 1];
K0 = [-1 -1; -1 -1];            % 初始化控制器矩阵 K
goal = [-5 -3 -1];              % 为闭合环路的特征值（极点）设置目标值向量
weight = abs(goal)           % 设置权值向量
lb = -4*ones(size(K0));       % 设置控制器的下界
ub = 4*ones(size(K0));       % 设置控制器的上界
options = optimset('Display','iter');    % 显示每次迭代的输出
[K,fval,attainfactor] = fgoalattain(fun,K0,goal,weight,[],[],[],[],lb,ub,[],...
options,A,B,D) % 计算多目标规划问题
```

输出如下：

```
weight =     5     3     1
                   Attainment                  Directional
 Iter   F-count      factor      Step-size      derivative    Procedure
   1      13          1.061          1             1.03
   2      20         0.4211          1           -0.679
   3      27       -0.06352          1            -0.523    Hessian modified
   4      34        -0.1571          1            -0.053    Hessian modified twice
   5      41        -0.3489          1            -0.133
   6      48        -0.3643          1          -0.00768    Hessian modified
   7      55        -0.3645          1        -4.25e-005    Hessian modified
   8      62        -0.3674          1          -0.00303    Hessian modified twice
   9      69        -0.3806          1           -0.0213    Hessian modified
  10      76        -0.3862          1           0.00266
  11      83        -0.3863          1        -2.73e-005    Hessian modified twice
  12      90        -0.3863          1        -1.21e-013    Hessian modified twice
Active Constraints:
     1
     2
     4
     9
    10
K =   -4.0000   -0.2564
```

```
      -4.0000   -4.0000
fval =   -6.9313
  -4.1588
       -1.4099
attainfactor =   -0.3863
```

函数 lsqlin 可以用来解决最小二乘最优问题，其完整调用格式为：

```
[x,resnorm,residual,exitflag, output,lambda] =
lsqlin(C,d,A,b,Aeq,beq,lb,ub,x0,options);
```

参数说明： 参数 C 和 d 是用来定义目标函数方程 Cx=d 的。resnorm 是 C*x-d 的 2-范数，即 norm(C*x-d) ^2。residual 是残差，即 residual=C*x-d。其他参数的意义同前面的函数。

例 10-13： 求下面系统的最小二乘解。

分析： 函数是 Cx=d，其中 C=magic(4)，d=[1:4]。约束为 "rand('state',0);A=rand(3,4); b=rand(3,1);" 且所有变量的上下界是 1 和-0.2。该问题的完整程序如下：

```
C = magic(4);              % 定义目标方程系数矩阵
d = [1:4];                 % 定义目标方程系数向量
rand('state',0);
A = rand(3,4);              % 定义约束条件
b = rand(3,1);              % 定义约束条件
lb = -0.1*ones(1,4);          % 定义下界
ub = 2*ones(1,4);           % 定义上界
[x,resnorm,residual,exitflag,output,lambda] = lsqlin(C,d,A,b,[ ],[ ],lb,ub) % 优化计算
```

上述程序的输出是

```
Optimization terminated successfully.
x =    0.1365
      0.3343
      -0.1000
      -0.1000
resnorm =    2.0333
residual =    0.2532
           0.5596
           -1.2313
          -0.3741
exitflag =     1
output = iterations: 4
       algorithm: 'medium-scale: active-set'
    firstorderopt: []
    cgiterations: []
lambda = lower: [4x1 double]
        upper: [4x1 double]
        eqlin: [0x1 double]
ineqlin: [3x1 double]
```

其中非线性不等式约束是否有效可以通过 lambda.ineqlin 取值查看。

10.3 利用极值画包络线

"包络线" 是一个高频调幅信号，它的幅度是按低频调制信号变化的。如果把高频调幅信号的

峰点连接起来，就可以得到一个与低频调制信号相对应的曲线，这条曲线就是（上）包络线。相应地把所有谷底的点按次序连起来就是（下）包络线。在很多振动剧烈的曲线的包络往往是节奏变化的曲线。在计算包络线的时候上面提到的峰值点和谷底点就分别是数据的极大值点和极小值点。

Wang Leii 于 2003 年发布了一个绘制包络线的程序，该程序实现了如图 10.8 所示中虚线框以外的部分。这里在该程序基础上增加了端点处理的补充，使得上下包络线完全夹住了数据。光盘中完整程序文件名是 envelopeL.m。

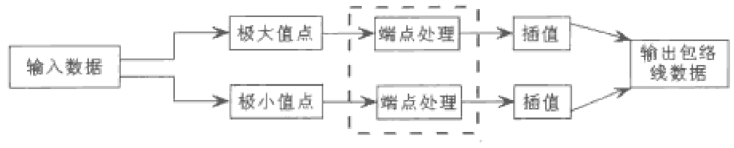


图 10.8　计算包络线的流程

如图 10.9 所示，一组变化剧烈的数据存在着一个明显的轮廓线。该曲线的函数表达式是 $y=\sin x\sin(20x)$。可以借助上面的 envelopeL 程序计算其上下包络线。这里只显示结果（如图 10.10 所示，其完整程序可以参阅光盘中的 envelopeL_test.m 文件），其中包络线用红色虚线标识。

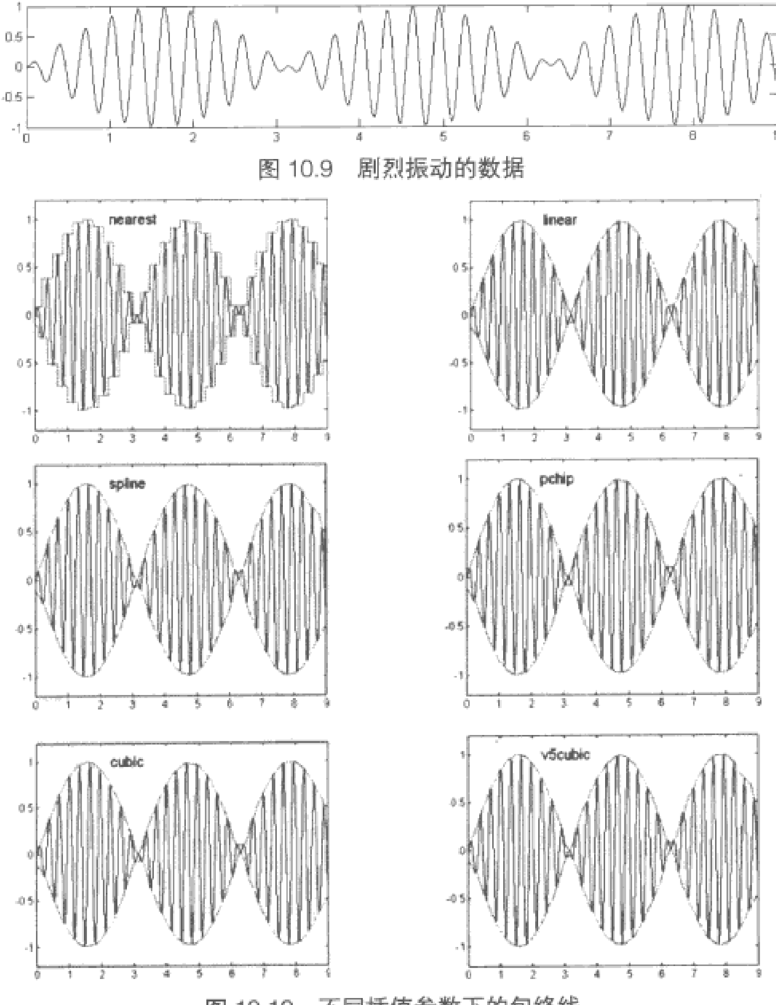


图 10.9　剧烈振动的数据

图 10.10　不同插值参数下的包络线

可以发现除了插值参数 nearest 下的包络线类似台阶状外，其他参数大体相同，在用的时候用户可以根据自己的喜好进行选择。接下来，考虑一种存在两侧极大值的情况。剧烈振荡的函数表达式如下：

$$x=\left|e^{-\frac{t^2}{40}}\sin 5t-\frac{1}{5}\right|$$

该函数的曲线如图 10.11 所示。

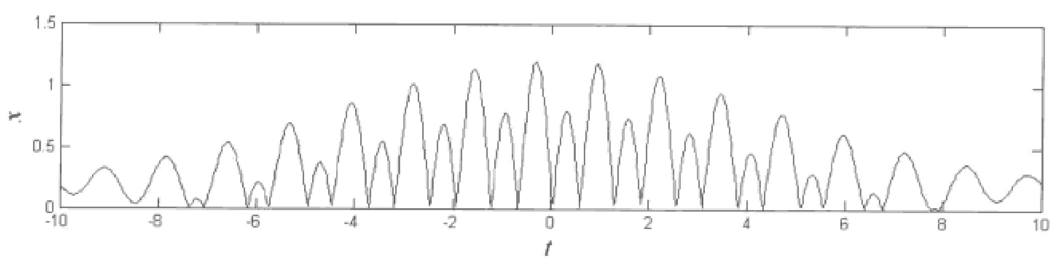


图 10.11 剧烈振荡的函数曲线

画出外层极大值对应的包络线，利用前面的 envelopeL 程序（使用插值参数 cubic）得到如图 10.12 所示的图形。

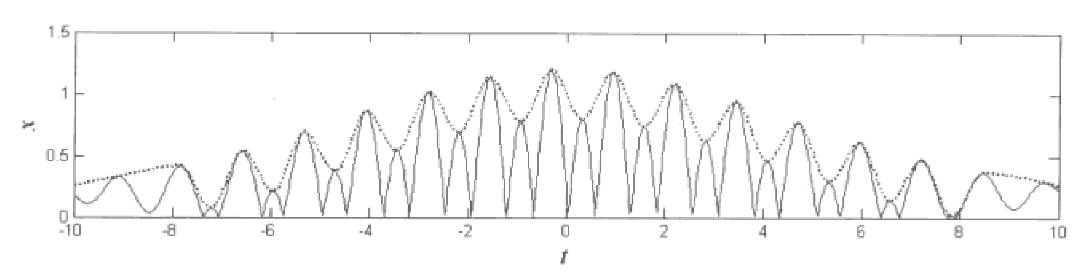


图 10.12 利用函数 envelopeL 得到的包络线

图 10.12 不是我们需要的结果。在图 10.11 中，一些极大值是不需要的，只需要最外层的极大值。而程序 envelopeL 只是根据所有极大值计算而返回上包络线，因此这个程序无法完成要求。为了剔除轮廓内部的极大值，可以使用计算两次极大值的思路处理（其中两端点处理使用一次函数进行拟合、插值而获得），所得结果如图 10.13 所示。

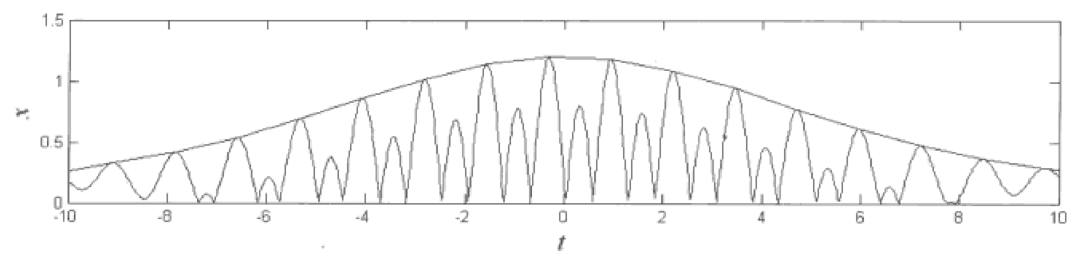


图 10.13 两次计算极大值而得到的包络线

如图 10.13 所示正是所期望的。相关计算由光盘中的 envelopeL2.m 文件完成。

10.4 小结

本章主要研究了离散数据和连续函数的最值问题。首先介绍了极大值和极小值的计算方法。其中连续函数的极值点可以通过计算函数的一阶（偏）导数获得，通过计算二阶（偏）导数来判断极大值还是极小值。离散情况极值的计算主要是通过差分代替导数来计算的，此外基于邻近数据的大小关系也可以确定出极值位置。然后介绍了最值的计算方法，离散数据的最值可以通过 MATLAB 中的两个基本函数 max 和 min 来计算。对于连续函数的最值问题，可以通过规划问题结合最优化工具箱来计算函数最小值，最大值可以通过简单的数学变换用求最小值的办法获取。最后结合极值的应用，介绍了离散数据包络线的计算方法。

◆ ◆ ◆

第 11 章　随机数的应用

本章包括

- **随机数的产生方法**　介绍不同分布的随机数、随机排序。
- **随机数的使用**　给出两个实例介绍使用方法。
- **统计量的计算**　介绍均值、方差、协方差和相关系数的计算。
- **回归分析**　介绍线性和非线性回归分析。

随机数在信号图像处理、统计物理和随机过程等问题中有着重要的应用。尽管计算机生成的随机数是一种伪随机数，它不是真正意义上的随机数，但是这样的伪随机数仍可认为是随机的，它具有不可预测性。本章对于各类分布形式的随机数给出了详细讲解，并且通过实际例子介绍随机数的使用。

11.1　随机数的产生

本节主要介绍产生不同概率分布的随机数及其使用方法。MATLAB 为使用者提供了多种随机函数，它们大多数存放在自带的统计工具箱之中，比如正态分布、Beta 分布和泊松分布等。统计工具箱被放置在路径\MATLAB6p5\toolbox\stats 中，用户可以进入这个路径查阅更多信息。

11.1.1　一般的随机函数调用格式

在 MATLAB 中使用随机数时，有时希望每次开启 MATLAB 之后，运行随机数的程序都得到同一个结果。为此建议大家首先设置一下随机函数的状态，即设定 state 的取值（MATLABr 5.0 版本之前称为 seed，之后的版本 seed 同样可以使用），具体方法如下：

```
rand('state',m);
randn('state',n);
```

这里 m 和 n 是随机函数的状态，它们可以是一个任意的数值，可以是 double 型数据，此如 1，2，3 等，也可以依据当前时钟数值取值，比如 sum(100*clock)。在实际应用中我们希望含随机数的程序每次开启 MATLAB 时运行结果都一样，所以应设置 m 和 n 为定值。



语句“rand('state',m);”可以控制除 randn 以外随机函数的状态，语句“randn('state',n);”可以控制除 rand 以外随机函数的状态。

在实际应用中，经常使用的随机函数是 rand，它产生 0~1 之间均匀分布的随机数。扩展到在任意两个数值之间产生均匀分布随机数的方法可以采用下面的程序：

```
a = 26;
```

```
b = 4;
rand('state',12);    % 设定状态为 12
x = a + (b-a) * rand(5)  % 产生 a 和 b 之间的随机数
```

输出如下：

```
x =   14.4156    5.5101   19.0885    7.8407   24.2386
      9.4955   11.0821   15.8734   18.4781    6.2287
     11.3599    4.3799   10.7317   10.4987    5.9698
      4.7910   12.4768   25.9685   24.9660    9.7228
     23.6205   13.2365   13.8334    5.8182   24.5396
```

这里 a 和 b 取值不等即可，没有大小关系限制。rand 函数可以用下面几种调用格式：

```
rand
rand(N)
rand(N,M)
rand(N,M,P,...)
```

它们依次产生 0 维（单个数值）、1 维、2 维和高维数组。产生正态分布（或者称为高斯分布）的函数是 randn，它的使用方法类似于 rand 函数，这里不再赘述。在 MATLAB 中，rand 和 randn 函数是开发技术人员嵌入 MATLAB 的函数，与其他随机函数不放在一起。对于一般的用户而言，没有特别需要统计工具箱可以不用安装。

在统计工具箱中，random 函数可以产生多种分布的随机数。其调用格式如下：

```
R = random('name',A,M,N);
```

其中 name 是分布的名称，A 是该分布的参数，M 和 N 表示生成矩阵的大小。其中 name 参数可以输入 Beta，Binomial，Chisquare，Exponential，F，Gamma，Geometric，Hypergeometric，Lognormal，Negative Binomial，Noncentral F，Noncentral t，Noncentral Chi-square，Normal，Poisson，Rayleigh，T，Uniform，Discrete Uniform，Weibull。

从这些单词可以看出相应分布的名称。在使用时可以仔细阅读相关帮助文档。通过测试，可以使用语句"rand('state',m)；"来控制 random 函数的状态，而"randn('state',m)；"不能控制 random 的状态。在实际应用中，确定随机数的状态还可以使用下面的方法完成，即用函数 save 把当前的随机数保存下来，以后使用该随机数时用 load 函数读入即可，但是这样需要产生一个多余的数据文件。

11.1.2 生成其他分布的随机函数

在 MATLAB 中还提供了一些专门随机函数，如表 11.1 所示，同时这里添加了关于相应分布的概率密度函数的数学公式。

表 11.1 特殊分布随机数列表

函数的调用格式	说明	概率密度函数
betarnd(A, B,m,n)	参数为 A 和 B 的 β 分布随机数	$f(x\mid a,b)=x^{a-1}(1-x)^{b-1}I_{0,1}(x)/B(a,b)$

（续表）

函数的调用格式	说明	概率密度函数
binornd(N,P,m,n)	参数为 N 和 p 的二项分布随机数	$f(x\mid n,p)=\binom{n}{x}p^{x}(1-p)^{n-x}I_{(0,1,\ldots,n)}(x)$
chi2rnd(N,m,n)	自由度为 N 的卡方分布随机数	$f(x\mid v)=x^{(v-2)/2}e^{-x/2}/\left[2^{v/2}\Gamma(v/2)\right]$
exprnd(mu,m,n)	参数为 mu 的指数分布随机数	$f(x\mid \mu)=\frac{1}{\mu}e^{-\frac{x}{\mu}}$
frnd(v_1, v_2,m,n)	第一自由度为 v_1、第二自由度为 v_2 的 F 分布随机数	$f(x\mid v_1,v_2)=\frac{\Gamma\left[\frac{v_1+v_2}{2}\right]}{\Gamma\left[\frac{v_1}{2}\right]\Gamma\left[\frac{v_2}{2}\right]}\left(\frac{v_1}{v_2}\right)^{\frac{v_1}{2}}\frac{x^{\frac{v_1-2}{2}}}{\left[1+\left(\frac{v_1}{v_2}\right)x\right]^{\frac{v_1+v_2}{2}}}$
gamrnd(a,b,m,n)	参数为 a 和 b 的 γ 分布随机数	$f(x\mid a,b)=\frac{1}{b^{a}\Gamma(a)}x^{a-1}e^{-\frac{x}{b}}$
geornd(p,m,n)	参数为 p 的几何分布随机数	$f(x\mid p)=p(1-p)^{x}I_{(0,1,\mathrm{K})}(x)$
hygernd(M,K,N,m,n)	参数为 M，K，N 的超几何分布随机数	$f(x\mid M,K,N)=\frac{\binom{K}{x}\binom{M-K}{N-x}}{\binom{M}{N}}$
lognrnd(mu, sigma,m,n)	参数为 mu 和 sigma 的对数正态分布随机数	$f(x\mid \mu,\sigma)=\frac{1}{x\sigma\sqrt{2\pi}}e^{\frac{-(\ln x-\mu)^{2}}{2\sigma^{2}}}$
nbinrnd(r,p,m,n)	参数为 r 和 p 的负二项式分布随机数	$f(x\mid r,p)=\binom{r+x-1}{x}p^{r}(1-p)^{x}I_{(0,1,\ldots)}(x)$

（续表）

函数的调用格式	说明	概率密度函数
ncfrnd(v₁,v₂,delta,m,n)	参数为 v_1，v_2，delta 的非中心 F 分布随机数	$f(x\mid v_1,v_2,\delta)=\sum_{k=0}^{\infty}\frac{e^{-\delta/2}(\delta/2)^k}{B(v_1/2,v_2/2+k)k!}\left(\frac{v_1}{v_2}\right)^{\frac{v_1}{2}+k}\times\left(\frac{v_2}{v_2+v_1x}\right)^{\frac{v_1+v_2}{2}+k}x^{v_1/2-1+k}$
nctrnd(v,delta,m,n)	参数为 v 和 delta 的非中心 T 分布随机数	$f(x\mid v,\delta)=\frac{v^{v/2}e^{-v\delta^2/2(x^2+v)}}{\sqrt{\pi}\Gamma(v/2)2^{(v-1)/2}(x^2+v)^{(v+1)/2}}$
ncx2rnd(v, delta,m,n)	参数为 v 和 delta 的非中心卡方分布随机数	$f(x\mid v,\delta)=\frac{1}{2}\mathrm{e}^{-(x+\delta)/2}\left(\frac{x}{\delta}\right)^{v/4-1/2}J_{v/2-1}\left(\sqrt{\delta x}\right)$
normrnd(mu,sigma,m,n)	参数为 mu 和 sigma 的正态分布随机数	$f(x\mid\mu,\sigma)=\frac{1}{\sigma\sqrt{2\pi}}e^{\frac{-(x-\mu)^2}{2\sigma^2}}$
poissrnd(lambda,m,n)	参数为 lambda 的泊松分布随机数	$f(x\mid\lambda)=\frac{\lambda^x}{x!}e^{-\lambda}I_{(0,1,\ldots)}(x)$
raylrnd(b,m,n)	参数为 b 的瑞利分布随机数	$f(x\mid b)=\frac{x}{b^2}e^{-\frac{x^2}{2b^2}}$
trnd(v,m,n)	自由度为 v 的 T 分布随机数	$f(x\mid v)=\frac{\Gamma\left(\frac{v+1}{2}\right)}{\Gamma\left(\frac{v}{2}\right)}\frac{1}{\sqrt{v\pi}}\frac{1}{\left(1+\frac{x^2}{v}\right)^{\frac{v+1}{2}}}$
unidrnd(N,m,n)	参数为 N 的均匀分布（离散）随机数	$f(x\mid N)=\frac{1}{N}I_{(1,\ldots,N)}(x)$

（续表）

函数的调用格式	说明	概率密度函数
unifrnd (a,b,m,n)	区间[a,b]上均匀分布（连续）随机数	$f(x\mid a,b)=\frac{1}{b-a}I_{[a,b]}(x)$
weibrnd(a,b,m,n)	参数为 a 和 b 的威布尔（韦伯）分布随机数	$f(x\mid a,b)=abx^{b-1}e^{-ax^b}I_{(0,\infty)}(x)$



函数 $I_{(0,1)}(x)$ 确保在区间[0,1]内 x 有非零概率值。$\Gamma(x)$ 是 Gamma 函数。$J_{v/2-1}\left(\sqrt{\delta x}\right)$ 是第一类贝塞尔函数。

表 11.1 中所列函数的使用类似于 rand 函数，这些函数的状态亦可用“rand('state',m);”来设定。这些分布往往带有随机分布参数，在使用之前需要阅读文档，使随机数正确产生。

其中 unidrnd 可以用于生成随机整数，而函数 unifrnd 可以产生指定区间内的随机数，即 unifrnd(A, B, M, N)产生 A 和 B 之间的随机数，这里要求 A<B。如果 A>B 系统会默认为错误，返回 NaN。例如使用 unifrnd(40,26,3,5)将会得到下面的结果：

```
ans =    NaN    NaN    NaN    NaN    NaN
       NaN    NaN    NaN    NaN    NaN
       NaN    NaN    NaN    NaN    NaN
```

11.1.3 随机排序函数类型

在 MATLAB 中，randperm 函数可以用来生成 1 到 N 的一个随机序列，如：

```
randperm(9)
ans =
    2     9     4     8     6     3     7     5     1
```

同样地，randperm 函数的状态需要用语句“rand('state',m);”来控制。利用这个函数可以对向量进行随机排序，下面是一段示例程序。

```
A = [2,3,7,4,5,6,8,9,10];
randn('state',1); %固定随机数的状态
se = randperm(length(A));
Ap = A(se)
```

输出结果是：

```
Ap =    10     8     4     7     6     3     9     5     2
```

Ap 是 A 的一个随机排列。类似地，可以利用这个函数对矩阵的行或者列的顺序进行随机排列。

下面的程序可以根据 se 来恢复 Ap 到原来的向量 A。

```
[sp,Ik] = sort(se);  % 对 se 按从小到大顺序排列
Ar = Ap(Ik)          % 用索引 Ik 把 Ap 恢复到原来的顺序
```

输出结果是：

```
Ar =    2    3    7    4    5    6    8    9    10
```

可以发现向量 Ar 与 A 是相等的。这里使用 sort 函数来获得恢复至原来顺序的次序向量 Ik。读者可以根据上面的方法对自己的数据进行排序处理。此外使用 randperm 函数还可以实现从 M 个数中选择 N 个数的目的，如：

```
rand('state',121);      % 设定随机函数的状态
A = rand(1,8)          % 产生 1×8 的随机数
ch1 = randperm(8);     % 产生 1:8 的一个随机排列
ch1 = ch1(1:2);        % 把 ch1 的前两个数赋值给 ch1
ch2 = randperm(8);     % 产生 1×8 的随机数
ch2 = ch2(1:5);        % 把 ch2 的前 5 个数赋值给 ch2
B1 = A(ch1),           % 从 A 中取出 ch1 代表位置处的 2 个数
B2 = A(ch2),           % 从 A 中取出 ch2 代表位置处的 5 个数
```

输出结果是：

```
A =    0.6080    0.0238    0.2023    0.3817    0.1651    0.8335    0.1721    0.2784
B1 =    0.3817    0.1651
B2 =    0.1721    0.6080    0.3817    0.2784    0.8335
```

上面的程序实现从向量 A 中随机地取出 2 个和 5 个数来。

11.1.4 计算概率密度函数的 MATLAB 函数

给定 x 是随机变量，如果存在一个非负函数 $f(x)$，使得对任意实数 $a, b(a<b)$ 有 $P(a<X\le b)=\int_a^b f(x)\mathrm{d}x$，则称 $f(x)$ 为 x 的概率密度函数。$f(x)$ 可以用来表示随机数的数字特征。概率密度函数具有下面两个性质：

- ◆ $f(x)\ge 0$
- ◆ $\int_{-\infty}^{+\infty} f(x)\mathrm{d}x=1$

利用概率密度函数的第 2 个性质可以计算出概率密度函数中的未知参数。概率密度函数能够给出随机数或者数据取不同幅值大小的概率，在随机振动、随机疲劳实验等应用场合，常常利用它来检测采集数据的正态性及了解幅值大小分布情况。

MATLAB 提供的计算概率密度函数列表如表 11.2 所示。

表 11.2 计算概率密度函数的 MATLAB 函数

调用形式	说明
betapdf(x, a, b)	参数为 a 和 b 的 β 分布概率密度函数值

（续表）

调用形式	说明
binopdf(x, n, p)	参数为 n 和 p 的二项式分布概率密度函数值
chi2pdf(x, n)	自由度为 n 的卡方分布概率密度函数值
exppdf(x, Lambda)	参数为 Lambda 的指数分布概率密度函数值
fpdf(x, n_1, n_2)	第一自由度为 n_1、第二自由度为 n_2 的 F 分布概率密度函数值
gampdf(x, a, b)	参数为 a 和 b 的 γ 分布概率密度函数值
geopdf(x, p)	参数为 p 的几何分布概率密度函数值
hygepdf(x, M, K, N)	参数为 M，K，N 的超几何分布概率密度函数值
lognpdf(x, mu, sigma)	参数为 mu 和 sigma 的对数正态分布概率密度函数值
nbinpdf(x, R, P)	参数为 R 和 P 的负二项式分布概率密度函数值
ncfpdf(x, n_1, n_2, delta)	参数为 n_1，n_2，delta 的非中心 F 分布概率密度函数值
nctpdf(x, n, delta)	参数为 n 和 delta 的非中心 t 分布概率密度函数值
ncx2pdf(x, n, delta)	参数为 n 和 delta 的非中心卡方分布概率密度函数值
normpdf(x, mu, sigma)	参数为 mu 和 sigma 的正态分布概率密度函数值
poisspdf(x, Lambda)	参数为 Lambda 的泊松分布的概率密度函数值
raylpdf(x, b)	参数为 b 的瑞利分布概率密度函数值
tpdf(x, n)	自由度为 n 的 T 分布概率密度函数值
unidpdf(x, n)	均匀分布（离散）概率密度函数值
unifpdf (x, a, b)	[a, b]上均匀分布（连续）概率密度在 x 处的函数值
weibpdf(x, a, b)	参数为 a 和 b 的威布尔（韦伯）分布概率密度函数值

此外还可以通过下面的格式计算某种分布的概率密度函数：

```
Y = pdf(name, X, A);
```

其中 name 是分布的名称，X 是变量矩阵，A 是分布的参数。详细用法可以参考 pdf 函数的帮助文档。

11.1.5 累积概率值

累积概率值可以用来计算产品或者试样在规定条件（或者规定参数）下符合要求和不符合要求的概率。因此借助这个参数可以来检查产品的生产情况、比较方案优劣、确定项目可行性和风险程度，MATLAB 提供的计算累积概率值的函数如表 11.3 所示。

表 11.3 专用函数的累积概率值函数表

调用形式	说明
betacdf(x, a, b)	参数为 a 和 b 的 β 分布累积分布函数值 $F(x)=P\{X\leq x\}$
binocdf(x,n,p)	参数为 n 和 p 的二项分布的累积分布函数值 $F(x)=P\{X\leq x\}$
chi2cdf(x, n)	自由度为 n 的卡方分布累积分布函数值 $F(x)=P\{X\leq x\}$
expcdf(x, Lambda)	参数为 Lambda 的指数分布累积分布函数值 $F(x)=P\{X\leq x\}$
fcdf(x, n_1, n_2)	第一自由度为 n_1、第二自由度为 n_2 的 F 分布累积分布函数值
gamcdf(x, a, b)	参数为 a 和 b 的 γ 分布累积分布函数值 $F(x)=P\{X\leq x\}$
geocdf(x,p)	参数为 p 的几何分布累积分布函数值 $F(x)=P\{X\leq x\}$
hygecdf(x,M,K,N)	参数为 M，K，N 的超几何分布累积分布函数值
logncdf(x, mu, sigma)	参数为 mu 和 sigma 的对数正态分布累积分布函数值
nbincdf(x, R, P)	参数为 R 和 P 的负二项式分布概累积分布函数值 $F(x)=P\{X\leq x\}$
ncfcdf(x, n_1, n_2, delta)	参数为 n_1，n_2，delta 的非中心 F 分布累积分布函数值

（续表）

调用形式	说明
nctcdf(x, n, delta)	参数为 n 和 delta 的非中心 T 分布累积分布函数值 F(x)=P{X≤x}
ncx2cdf(x, n, delta)	参数为 n 和 delta 的非中心卡方分布累积分布函数值
normcdf(x, mu, sigma)	参数为 mu 和 sigma 的正态分布累积分布函数值 F(x)=P{X≤x}
poisscdf(x,Lambda)	参数为 Lambda 的泊松分布累积分布函数值 F(x)=P{X≤x}
raylcdf(x, b)	参数为 b 的瑞利分布累积分布函数值 F(x)=P{X≤x}
tcdf(x, n)	自由度为 n 的 T 分布累积分布函数值 F(x)=P{X≤x}
unidcdf(x,n)	均匀分布（离散）累积分布函数值 F(x)=P{X≤x}
unifcdf (x, a, b)	[a,b]上均匀分布（连续）累积分布函数值 F(x)=P{X≤x}
weibcdf(x, a, b)	参数为 a 和 b 的威布尔（韦伯）分布累积分布函数值 F(x)=P{X≤x}



累积概率函数就是分布函数 F(x)=P{X≤x}在 x 处的值。

此外还可以通过下面的格式计算某种分布的累积概率函数：

```
Y = cdf(name, X, A);
```

其中 name 是分布的名称，X 是变量矩阵，A 是分布的参数。详细用法可以参考 cdf 函数的帮助文档。函数 ecdf 可以用来计算经验（Kaplan-Meier）累计分布函数。

11.1.6 逆累积分布函数

在已知概率值求该概率的分布点时，就要用到逆累积分布函数。利用这个函数，可以根据概率值计算相关参数的取值，从而对测试对象的性能等指标进行有效的估计。MATLAB 提供的相关函数如表 11.4 所示。

表 11.4 逆累积分布函数

调用形式	说明
betainv (x, a, b)	β 分布逆累积分布函数
binoinv (x, n, p)	二项分布逆累积分布函数
chi2inv (x, n)	卡方分布逆累积分布函数
expinv (p, Lambda)	指数分布逆累积分布函数
finv (x, n_1, n_2)	F 分布逆累积分布函数
gaminv (x, a, b)	γ 分布逆累积分布函数
geoinv (x, p)	几何分布逆累积分布函数
hygeinv (x, M, K, N)	超几何分布逆累积分布函数
icdf(NAME, p, a, b)	计算逆累积分布函数，其用法类似于 cdf
logninv (x, mu, sigma)	对数正态分布逆累积分布函数
nbininv (x, R, P)	负二项式分布逆累积分布函数
ncx2inv (x, n, delta)	非中心卡方分布逆累积分布函数
ncfinv (x, n_1, n_2, delta)	非中心 F 分布逆累积分布函数
nctinv (x, n, delta)	非中心 T 分布逆累积分布函数
norminv(x, mu, sigma)	正态分布逆累积分布函数
poissinv (x, Lambda)	泊松分布的逆累积分布函数

（续表）

调用形式	说明
raylinv (x, b)	瑞利分布逆累积分布函数
tinv (x, n)	T 分布累积分布函数
unidinv (p, n)	均匀分布（离散）逆累积分布函数，x 为临界值
unifinv (p, a, b)	均匀分布（连续）逆累积分布函数（P=P{X≤x}，求 x）
weibinv (x, a, b)	威布尔（韦伯）分布逆累积分布函数

通过下面的格式计算某种分布的逆累积概率函数：

```
Y = icdf (name, X, A);
Y = icdf (name, X, A, B);
```

其中 name 是分布的名称，X 是变量矩阵，A 和 B 是随机分布的参数。具体使用一个参数还是两个参数需要根据分布来确定。使用者可以根据表 11.4 进行选择。

11.2　随机数的使用

在这一节中，结合实际问题给出随机数使用的例子。其中 Galton 板实验是二项分布的一个经典的例子，而赌徒输光问题是人们早期进行概率问题研究的实例。

11.2.1　Galton 板实验

例 11–1：模拟 Galton 板实验。

一个 8 级 Galton 板实验系统如图 11.1 所示（该图由作者编写的 Galton_board.m 和 Galton_board2.m 程序完成）。在图(a)中，当小球从顶部向下降落时，遇到第一层竖隔板，此时小球向左落下和向右落下的概率各占一半，即小球向左和向右运动的概率为 0.5；当小球继续下落遇到第二层竖隔板时，小球向左和向右的概率仍是 0.5，以后每一层情况都是如此（一个可能的过程如图(b)所示）。最后到了第 8 层底部，小球将落入底部 9 个槽中的其中一个。但是小球究竟落入哪一个槽内的概率是不一样的。如果将这 9 个槽编号为 1，2，3，4，5，6，7，8，9，计算小球落入 9 个槽中的概率。

理论分析：这是一个经典的二项分布概率模型。考虑在第 k 层小球运动方向有两种可能，用 X_k 表示，则 X_k 服从两点概率分布。这里用 $X_k=1$ 表示竖隔板向右侧运动，用 $X_k=0$ 表示竖隔板向左侧运动，它们发生的可能性都是 50%。

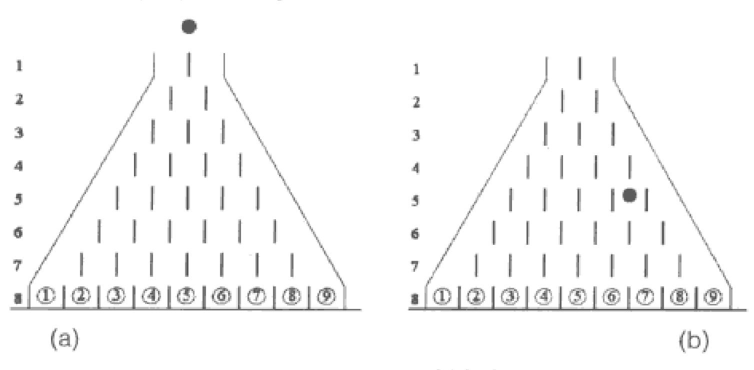


图 11.1　Galton 板实验

最终位置 X 由下式决定：

$$X = \sum_{k} X_k$$

可知 X 满足二项分布，即 $X \sim B(8,0.5)$。该二项分布的概率密度可用下面的程序获得：

```
bpdf = binopdf(0:8,8,0.5);
```

接下来用 MATLAB 模拟小球下落到 9 个槽内的概率分布。在各层中，用 0 和 1 分别表示向左和向右运动。在小球下落到底层的槽内时，一个 8 位的二进制数就完全确定了。而所落入槽的编号应该是这个 8 位二进制数各数位按十进制数相加的结果。重复模拟小球下落过程 8000 次，可以统计出小球落入各槽内的次数（即频数）。画出 9 个频数数据的直方图，如图 11.2 所示。相应程序如下：

```
rand('state',0)                  % 固定随机数产生状态
R = unidrnd(2,8,8000)-1;         % 产生 1~7 的随机整数
test = sum(R);                  % 对随机数求和
h = hist(test,9);                % 统计各数出现的频数
bpdf = binopdf(0:8,8,0.5);       % 计算二项分布的理论值
bpdf = bpdf/max(bpdf)*max(h);  % 使模拟数据与理论数据保持量级一致
b1 = bar(1:9,[h;bpdf]');         % 绘制直方图
set(b1(2),'facecolor','w');       % 设置右侧的直条填充颜色为白色
set(b1(1),'facecolor',[0.4,0.4,0.4]);   % 把左侧直条填充颜色设置为灰色
set(gca,'Position',[0.13 0.51 0.775 0.415]);   % 重置坐标轴位置
```

上面程序运行结果如图 11.2 所示。

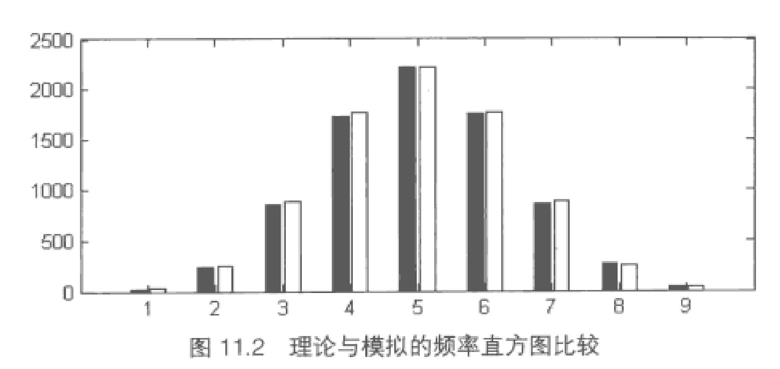


图 11.2 理论与模拟的频率直方图比较

在图 11.2 中，灰色线条表示模拟值，而白色线条表示理论值（其中把概率理论值的最大值调整为模拟值的最大值以便于比对，该程序的名称是 simulation11_1.m），可以发现二者符合得很好。

11.2.2 赌徒输光问题

例 11–2：赌徒输光问题。

两个赌徒甲、乙将进行一系列赌博。在每一局中甲获胜的概率为 p，而乙获胜的概率为 q，p+q=1。在每一局后，失败者都要付一元钱给胜利者。在开始时甲拥有资本 a 元，而乙有资本 b 元，两个赌徒直到甲输光或乙输光而停止赌博。求甲输光所有钱的概率。

通过理论分析可知甲输光所有钱的概率是：

$$P=\begin{cases}\dfrac{b}{a+b}, & p=\dfrac{1}{2}\\ \dfrac{1-g^b}{1-g^{a+b}}, & p\neq\dfrac{1}{2}\end{cases},\quad g=\frac{p}{q}$$

模拟赌博过程的思路：在每一次模拟中，随机地产生 1 个数，如果该数小于 p，说明赌徒甲获胜，相应甲得到一元钱，而乙付出一元钱；反之甲拿出一元钱给乙。这里对甲的赌资 a、乙的赌资 b、甲赢得概率 p 取不同数值进行 10000 次赌博过程模拟，相应程序如下：

```
a=10;
b=3;
p=0.55;
S=0;                % 计数器置 0
N=10000;             % 模拟的次数
m=6;                 % 设定随机数状态值，改变这个值可以进行不同的实验
rand('state', m);       % 设置随机数状态
for k=1:N;
    at=a;            % 初始化甲的赌资
    bt=b;            % 初始化乙的赌资
    while at>0.5&bt>0.5;      % 模拟整个赌博过程
        r=[(rand<p)-0.5]*2;    % 算输赢
        at = at+r;            % 交换赌资
        bt = bt-r;             % 交换赌资
    end
    S=S+(at<0.5);             % 若甲输，累加甲输的次数
end
P=S/N                        % 计算甲输的概率值
g=p/[1-p];
Po=[1-g^b]/[1-g^(a+b)]            % 返回甲输掉所有赌资概率的理论值
```

上述程序对应的名字是 robbers.m。所有计算结果如表 11.5 所示。

表 11.5 赌徒输光问题的多次模拟结果

参数与理论值		m					
a,b,p	Po	1	2	3	4	5	6
10,3,0.55	0.0656	0.0648	0.0648	0.0612	0.0676	0.0672	0.0638
5, 9, 0.6	0.1287	0.1227	0.1338	0.1207	0.1272	0.1269	0.1262
5, 7, 0.56	0.2584	0.2583	0.2628	0.2522	0.2599	0.2616	0.2555
3,18,0.58	0.3790	0.3875	0.3846	0.3731	0.3890	0.3764	0.3724
4,15,0.55	0.4357	0.4313	0.4414	0.4347	0.4391	0.4346	0.4322
3, 9, 0.54	0.5529	0.5546	0.5536	0.5439	0.5566	0.5543	0.5473
8, 3, 0.41	0.6768	0.6794	0.6721	0.6703	0.6777	0.6836	0.6717
9, 3, 0.38	0.7719	0.7732	0.7728	0.7734	0.7706	0.7762	0.7737
9, 6, 0.43	0.8278	0.8274	0.8307	0.8252	0.8300	0.8313	0.8192

11.3 统计量的计算

本节介绍一些确定随机数的特征参数，包括均值、方差、协方差和相关系数等。它们对于随机

数的特征描述和数据分析等具有重要的意义。

11.3.1 单值参数

在一些关于随机数或者实验数据分析的时候，往往需要知道这些数据的平均值和方差等于多少，以便和预期的进行比较。本小节将介绍相关参数的计算。

11.3.1.1 期望值

函数 mean，mean2，nanmean，median，nanmedian，geomean，harmmean 的用法介绍如下。

函数 mean 有如下几种常用的调用方法，相应的返回值都是平均值，只是输入量有一定的差别。

```
mean(X)        % 返回向量 X 的均值
mean(A)        % 返回矩阵 A 中各列的平均值
mean(A,dim)    % 在给出多维数组 A 相应维数内的平均值
```

如果需要计算矩阵的均值，可以调用格式为 mean2(A)。

Nanmean 函数类似于 mean 函数，只是忽略了 nan 的计算。如：

```
>> nanmean([3,nan,1])
ans =     2
```

在这个函数中把 nan 去掉，不计入统计平均。

如果需要返回所求数据的中值（中位数），就可以调用函数 median，如下所示：

```
>> A=[ 3 14 5; 3 2 4 6; 3 1 1 5]
A =      3     1     4     5
       3     2     4     6
       3     1     1     5
>> nanmedian(A)
ans =     3     1     4     5
```

函数 nanmedian 的用法和意义分别类似于函数 median 和 nanmean。

geomean 函数用于计算几何平均数，几何平均数的数学含义是：$\left(\prod_{k=1}^{n} x_k\right)^{1/n}$，其中 n 表示数据的个数。该函数的用法同 mean 函数。

harmmean 函数计算调和平均值，调和平均值的数学含义是：$\dfrac{n}{\sum_{k=1}^{n}\dfrac{1}{x_k}}$，其中 n 表示数据的个数。该函数的用法同 mean 函数。

11.3.1.2 最大值与最小值之差

range(...)返回数据中最大值与最小值之差，用法类似于 mean 函数。该函数等价于 max(A)-min(A)。

11.3.1.3 方差

方差的数学定义为：$\frac{1}{n-1}\sum_{k=1}^{n}(x_k-\bar{X})^2$，其中 n 是数据的个数，$\bar{X}$ 是相应的均值。函数 var 的使用方法如下：

```
D = var(X)       %返回向量 X 的方差
D = var(A)       %返回矩阵 A 各列向量的方差
D = var(X, 1)    %返回向量（矩阵）X 的简单方差（即置前因子为1/n 的方差）
D = var(X, w)    %返回向量（矩阵）X 的以 w 为权重的方差
```

11.3.1.4 标准差

标准差的数学定义是：

$$\sqrt{\frac{1}{n-1}\sum_{k=1}^{n}\left(x_k-\bar{X}\right)}$$

函数 std 使用方法如下：

```
std(X)              % 返回向量（矩阵）X 的样本标准差（置前因子为1/(n-1)）
std(X,1)            % 返回向量（矩阵）X 的标准差（置前因子为1/n ）
std(X, 0)           % 与 std(X)作用相同
std(X, flag, dim)   % 返回向量（矩阵）中维数为 dim 的标准差值，其中 flag=0 时，置前因子为
1/(n-1)，否则置前因子为1/n
```

类似地，函数 nanstd 计算忽略 NaN 的标准差。

11.3.1.5 样本的偏斜度

偏斜度的数学定义为：

$$\frac{\mathrm{E}(x-\mu)^3}{\sigma^3}$$

样本数据的偏斜度是关于均值不对称的一个测度，如果偏斜度为负，说明均值左边的数据比均值右边的数据更散；如果偏斜度为正，说明均值右边的数据比均值左边的数据更散，所以正态分布的偏斜度为 0。函数 skewness 的用法是：

```
y = skewness(X)         % 返回向量 X 的偏斜度
y = skewness(A)         % 返回矩阵 A 各列的偏斜度
y = skewness(X,flag)    % flag=0 表示偏斜纠正，flag=1（默认）表示偏斜不纠正
```

11.3.1.6 特殊分布的均值和方差

MATLAB 提供了相关函数计算分布的均值和方差，这些函数一般的调用格式为：

```
[M, V] = functionname(P)  % M是均值，V是方差，P表示分布的参数
```

具体函数的使用如表 11.6 所示。

表 11.6 特殊分布的均值和方差

调用格式	说明	均值	方差
betastat (a, b)	β分布	$\frac{a}{a+b}$	$\frac{ab}{(a+b+1)(a+b)^2}$
binostat (n, p)	二项分布	np	$np(1-p)$
chi2stat (x, n)	卡方分布	n	$2n$
expstat (p, mu)	指数分布	μ	μ^2
fstat (nu1, nu2)	F 分布	$\frac{v_2}{v_2-2}$	$\frac{2v_2^2(v_1+v_2-2)}{v_1(v_2-2)^2(v_2-4)}$
gamstat (a, b)	γ分布	ab	ab^2
geostat (p)	几何分布	$(1-p)/p$	$(1-p)/p^2$
hygestat (M,K,N)	超几何分布	$\frac{NK}{M}$	$N\frac{K}{M}\frac{M-K}{M}\frac{M-N}{M-1}$
lognstat (mu, sigma)	对数正态分布	$e^{\mu+\frac{\sigma^2}{2}}$	$e^{2\mu+2\sigma^2}-e^{2\mu+\sigma^2}$
nbinstat (r, p)	负二项式分布	$r(1-p)/p$	$r(1-p)/p^2$
ncfstat (nu1, nu2, delta)	非中心F分布	$\frac{v_2(\delta+v_1)}{v_1(v_2-2)}$, $v_2>2$	$2\left(\frac{v_2}{v_1}\right)^2\left[\frac{(\delta+v_1)^2+(2\delta+v_1)(v_2-2)}{(v_2-2)^2(v_2-4)}\right]$
nctstat (n, delta)	非中心T分布	$\frac{\delta(v/2)^{1/2}\Gamma((v-1)/2)}{\Gamma(v/2)}$ $v>1$	$\frac{v}{v-2}(1+\delta^2)-\frac{v}{2}\delta^2\left[\frac{\Gamma((v-1)/2)}{\Gamma(v/2)}\right]^2$ $v>2$
ncx2stat (nu, delta)	非中心卡方分布	$v+\delta$	$2(v+2\delta)$

（续表）

调用格式	说明	均值	方差
normstat(mu, sigma)	正态分布	μ	σ^2
poisstat (lambda)	泊松分布	λ	λ
raylstat (b)	瑞利分布	$b\sqrt{\pi/2}$	$\frac{4-\pi}{2}b^2$
tstat (n)	T 分布	$0,\ v>1$	$v/(v-2),\ v>2$
unidstat (n)	均匀分布（离散）	$(a+b)/2$	$(b-a)^2/12$
unifstat (a, b)	均匀分布（连续）	$(a+b)/2$	$(b-a)^2/12$
weibstat (a, b)	威布尔（韦伯）分布	$a^{\frac{1}{b}}\Gamma(1+b^{-1})$	$a^{\frac{2}{b}}\left[\Gamma(1+2b^{-1})-\Gamma^2(1+b^{-1})\right]$

11.3.2 多值参数

在比较两组数据的相似或者相关程度的时候，我们需要对整个范围内的情况做一个了解，这时仅用一个参数是无法完成任务的，为此可以使用协方差和相关系数来描述。

11.3.2.1 协方差

协方差的数学描述是：$E\left[(x_1-\mu_1)(x_2-\mu_2)\right]$，其中 $\mu_1=Ex_1$，$\mu_2=Ex_2$。函数 cov 的用法如下：

```
cov(X)        % 返回向量 X 的协方差
cov(A)        % 返回矩阵 A 的协方差矩阵，该协方差矩阵的对角线元素是 A 的各列的方差
cov(X,Y)       % X 和 Y 是等长列向量，该语句等同于 cov([X,Y])
```

11.3.2.2 相关系数

相关系数的数学定义为 $\text{corrcoef}=\dfrac{C_{m,n}}{\sqrt{C_{m,m}}\sqrt{C_{n,n}}}$，$C=\text{cov}(X)$。

函数 corrcoef 的调用格式如下：

```
corrcoef (A)       % 返回矩阵 A 的列向量的相关系数矩阵
corrcoef (X,Y)     % 返回列向量 X 和 Y 的相关系数，其等同于 corrcoef([X,Y])
```

11.4 回归分析

回归分析是研究变量之间相关关系的一种统计方法，即利用统计数据寻求变量间关系的近似表达式（或者称为经验公式），同时使用所得公式进行统计描述、分析和推导，以解决预测、优化和控制等问题。回归分析可以分为线性回归和非线性回归两种类型。

MATLAB 提供的回归分析函数如表 11.7 所示。

表 11.7 回归分析函数表

分类	函数名	说明
线性回归	regeress	多元线性回归
	regstats	回归统计量诊断
	ridge	岭回归
	rcoplot	绘制残差及其置信区间曲线
	rstool	多维响应面可视化
	robustfit	稳健回归模型拟合
	stepwise	交互式逐步回归
	x2fx	用于设计矩阵的因子设置矩阵
非线性回归	nlinfit	非线性最小二乘数据拟合（牛顿法）
	nlintool	非线性模型拟合的交互式图形工具
	nlparci	参数的置信区间
	nlpredci	预测值的置信区间
	nnls	非负最小二乘

11.4.1 线性回归

线性回归包括一元线性回归和多元线性回归，这种回归关于未知参数和回归变量都是线性关系，其变量关系可以表示为：

$$Y=\beta_0+\beta_1X_1+\beta_2X_2+\cdots+\beta_nX_n+\varepsilon=\varepsilon+\beta_0+\sum_{k=1}^{n}\beta_kX_k$$

其中 β_k 被称为回归系数，ε 是随机误差，同时有 ε 的均值和方差分别是 $E(\varepsilon)=0$ 和 $D(\varepsilon)=\sigma^2$。当 $m=1$ 时上述问题为一元线性回归，当 $m>1$ 时为多元线性回归。在 MATLAB 中提供了一个多元线性回归函数 regress，该函数的调用格式为：

```
[b, bint, r, rint, stats] = regress(y, x, a);
```

输入参数说明： y 为观测到的随机变量矩阵，x 为自变量矩阵。若回归关系中包括常数项，那么 x 的第一列应该全部等于 1。x 和 y 应该有相同的行数，x 的列数等于参数的个数。当 x 为两列且第一列都等于 1 时为一元线性回归。A 是输出各种置信区间用到的显著水平。

输出参数说明： b 是参数的点估计。bint 是参数的区间估计。r 是残差的点估计。rint 是残差的

区间估计。当点估计落在区间估计之外时，拒绝无效假设。stats 中包含 3 项：第 1 项为回归方程的决定系数 R^2，第 2 项是回归方程的 F 统计量，第 3 项是拒绝无效假设的概率。

下面应用 regress 函数进行回归分析。

例 11-3：一元回归分析。

某连锁公司对其子公司的主要产品（各不相同）进行成本 x 与售价 y 分析，得到 10 个不同地点的分公司的相关数据（单位：元），如下：

售价 y	5.4	6.7	7.3	7.8	9.9	10.9	13.1	15.9	14.1	10.4
成本 x	1.5	2.1	3.2	4.6	7.5	9.2	10.2	12.1	11.1	8.4

在进行回归分析之前，可以把 x 和 y 的数据点用图形表示出来，如图 11.3 所示。可以看出“斜十字”大致沿一条直线分布，因此我们考虑进行线性回归分析。相应的回归分析程序如下：

```
x = [ones(10,1),[1.5,2.1,3.2,4.6,7.5,9.2,10.2,12.1,11.1,8.4]'];   % 输入 x 数据
y = [5.4,6.7,7.3,7.8,9.9,10.9,13.1,15.9,14.1,10.4]';              % 输入 y 数据
[b,bint,r,rint,s]=regress(y, x, 0.01);                             % 进行线性回归
b,s
```

相应的输出结果为：

```
b =    4.1071
       0.8645
s =0.9473  143.8458    0.0000
```

上面显示的计算结果表明，x 和 y 的决定系数达到了 $R^2=0.9473$，回归系数和误差都达到了显著水平，因而得到回归方程为 $y=4.1071+0.8645x$ 是可以信赖的。完整程序名称为 regression1.m，数据点和相应曲线如图 11.3 所示。

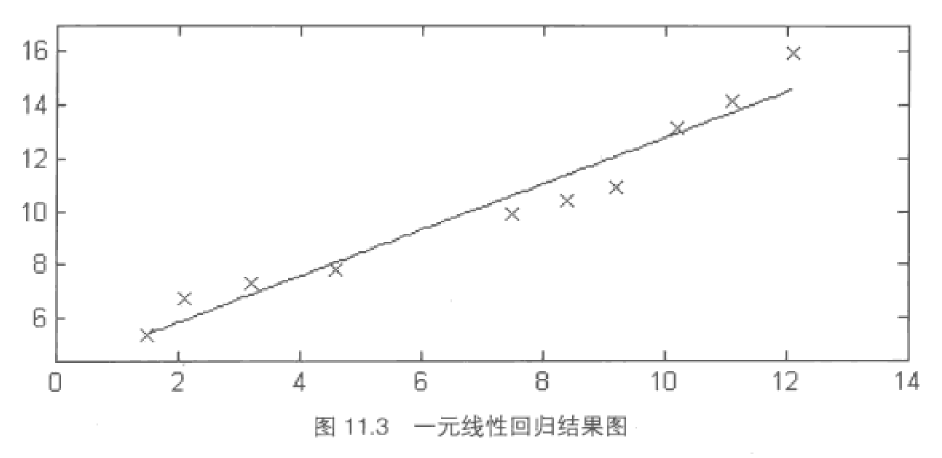


图 11.3 一元线性回归结果图

例 11-4：多元回归分析。

某研究所在一次实验中测定了土壤中成分 A 的含量 x1，成份 B 的含量 x2，成份 C 的含量 x3，同时得到某农作物吸收某养分的含量 y。进行 x 和 y 的回归分析，建立 x 和 y 之间的关系如表 11.8 所示。

表 11.8 多元回归分析相关数据

y	x1	x2	x3
64	0.4	53	158
60	0.4	23	163
71	3.1	19	37
61	0.6	34	157
54	4.7	24	59
77	1.7	65	123
81	9.4	44	46
93	10.1	31	117
93	11.6	29	173
51	12.6	58	112
76	10.9	37	111
96	23.1	46	114
77	23.1	50	134
93	21.6	44	73
95	23.1	56	168
54	1.9	36	143
168	26.8	58	202
99	29.9	51	124

在这个问题中，自变量 x 含有 3 个分量。x 和 y 之间的关系可能是线性的，也可能是非线性的。这里进行多元线性回归分析的尝试（非线性回归分析将在后面介绍），用下面的程序（完整程序文件名是 regression2.m）进行回归分析。

```
[b,bint,r,rint,s]=regress(y, x, 0.05);  % 进行多元线性回归分析
b=b', bint, s                          % 把数据 b 转置并显示，同时显示 bint 和 s
```

相应的结果输出为：

```
b =   43.6522    1.7848   -0.0834    0.1611
bint =  5.0241   82.2803
        0.6315    2.9380
        -0.9793    0.8125
        -0.0784    0.4006
s =    0.5493    5.6885    0.0092
```

根据上面的数据，可以写出回归方程为：

$$y = 43.6522 + 1.7848x_1 - 0.0834x_2 + 0.1611x_3$$

因为 $R^2=0.5493$，$F=5.6885$，$p=0.0092<0.01$，所以方程是极显著的，但未必每一个变量对 y 的贡献都是极显著的。

为了求出所有显著贡献的变量，MATLAB 还提供了一个逐步计算回归的函数 stepwise，该函数是交互图形工具函数，调用格式是：

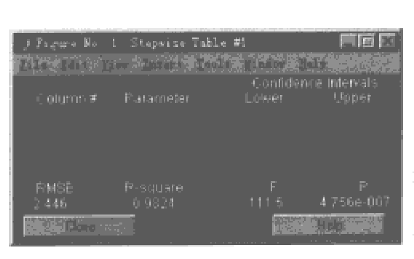
```
stepwise(x, y, inmodel, alpha),
```

输入参数说明： x 是自变量矩阵，其列数为自变量的个数。若回归中有常数项，则 x 的第 1 列都等于 1。y 表示函数值矩阵。inmodel 为输入模式，即由起始选择的自变量序号组成，如起始选择 x1 和 x3，那么 inmodel=[1, 3]。inmodel 的默认值是全部变量，即[1:size(x,2)-1]。alpha 是显著

水平，表示排除和选中变量用到的显著水平。执行该函数后将会弹出 3 个 Figure 窗口（如图 11.4 所示）：第 1 个是参数估计值和置信区间，决定系数 R^2 和统计量 F 等；第 2 个是逐步回归的历史图；第 3 个是逐步回归诊断表，凡是系数估计值距 0 刻度线很近且置信区间穿越 0 刻度线（其由绿色虚线表示），那么该系数与 0 无显著差异，属于可以排除的变量，反之远离 0 刻度线，同时置信区间不穿越 0 刻度线（此时其由白色实线表示），那么该系数与 0 有显著差异，该变量应该保留，直至留下的变量全部显著为止。

执行下面的语句：

```
load hald;                    % 读入 hald.mat 数据文件
stepwise(ingredients,heat)    %交互式的回归分析，ingredients 相当于自变量，heat 相当于函数值
```



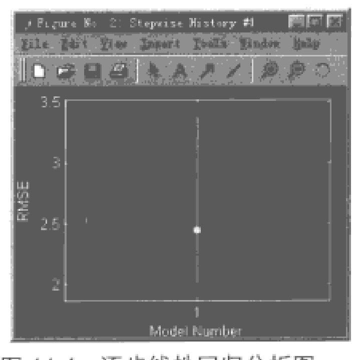




图 11.4　逐步线性回归分析图

11.4.2　非线性回归

很多实际问题往往是非线性的，这时需要进行非线性回归分析。它的数学描述是：

$$f(x,y,\beta)=\varepsilon$$

其中 f 是函数关系，x 是自变量（它代表一个或者多个量），y 函数变量，β 是待定的参数，ε 是随机误差。假设已知样本数据 (x_k,y_k)，其中 $1\le k\le K$，$x_k=\left[x_{k,1},x_{k,2},\dots,x_{k,n}\right]$（$n\ge1$，表示 x 中包含参量的个数），那么可以得到下面的系列方程：

$$f(x_k,y_k,\beta_k)=\varepsilon_k$$

在实际问题中，最常见的非线性回归模型为 x 和 y 是可分离变量的，即：

$$f(x,y,\beta)=y-g(x,\beta)=\varepsilon$$

这里 $g(x,\beta)$ 被称为期望函数，在回归分析中需要根据物理意义预先定义，它可以是多项式函数、分式、指数函数以及三角函数等。

在 MATLAB 统计工具箱中提供了 nlinfit 函数来实现非线性回归分析，其调用格式为：

```
[beta, r, j] = nlinfit(x, y, modelname, beta0);
```

参数说明： 其中 beta 是返回的回归系数，r 是残差，j 是 Jacobian 矩阵。输入数据 x 和 y 分别是 m×n 矩阵和 n 维列向量，modelname 是回归模型的数学表达式，beta0 是预先设定的回归系数初值。

例 11-5：一元非线性回归。

根据下面的数据进行一元非线性回归分析：

x	1	2	3	4	5	6
y	0.39	0.44	0.45	0.46	0.47	0.48

可以发现上面的数据之间不满足线性关系，相关数据点如图 11.4 所示。对于这样的数据分布可以用多种非线性函数作为模型。在这里选择的非线性模型是：

$$y = g(x,\beta) = \frac{\beta_1 x}{\beta_2 + x}$$

使用下面的程序定义上面的非线性模型：

```
funm = inline('b(1)*x./[b(2)+x]', 'b', 'x');
```

定义非线性函数，也可以写一个函数文件来定义这个非线性函数，即：

```
function y=funm(b,x);
y =  b(1)*x./[b(2)+x];
```

函数 nlinfit 可以用来完成非线性回归分析，利用下面的语句可以完成本例子的非线性回归过程（完整程序是 regression3.m）。

```
x = 1:6;
y = [0.39, 0.45, 0.46, 0.47, 0.48, 0.49];
beta0 = [1, 1];
[beta,r,j] = nlinfit(x, y, funm, beta0);
beta, r
```

输出结果是：

```
beta =
    0.5103
    0.3028
r =
   -0.0017
    0.0068
   -0.0036
   -0.0044
   -0.0012
0.0042
```

相应的数据和回归曲线如图 11.5 所示。

对于上述问题我们还可以使用 nlintool 进行回归分析，执行 "nlintool(x, y, funm, beta0);" 语句，将会得到如图 11.6 所示的图形。

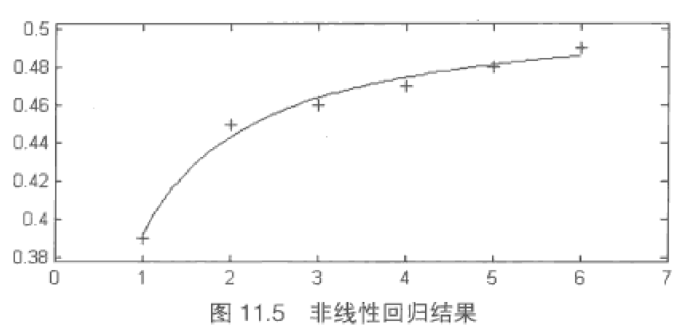


图 11.5 非线性回归结果

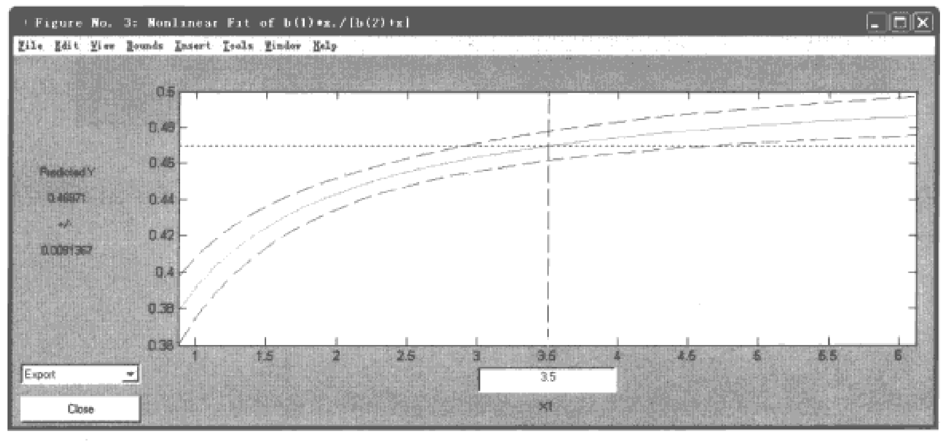


图 11.6 交互式回归分析

在图 11.6 中，蓝色虚线的位置可以用鼠标移动，同时也可以在 X1 上面的编辑框中输入数值来移动其位置。原始数据分布在两条红色曲线之间，在 Y 轴上实时显示当前点的函数值和残差（与红线比较）。

例 11–6： 多元非线性回归。

这里我们利用 MATLAB 提供的数据 reaction.mat 进行多元非线性回归分析。非线性模型是：

$$y = g(x,\beta) = \frac{\beta_1 x_2 - x_3 / \beta_5}{1 + \beta_2 x_1 + \beta_3 x_2 + \beta_4 x_3}$$

这个模型被称为 Hougen-Watson 模型，其中 $g(x,\beta)$ 被称为 hougen 函数。在实际问题中，可以根据各个参数之间的物理意义和参数对输出结果的影响来建立函数关系模型。

在 MATLAB 中 hougen.m 是计算上述公式的函数文件。

```
load reaction;              % 读入 reaction.mat 数据
betafit = nlinfit(reactants, rate, @hougen, beta)   % 进行多元非线性回归分析
nlintool(reactants, rate, @hougen, beta)         % 交互式多元非线性回归分析
```

输出结果是：

```
betafit =
    1.2526
    0.0628
    0.0400
    0.1124
1.1914
```

其中 nlintool 函数输出如图 11.7 所示的图形。

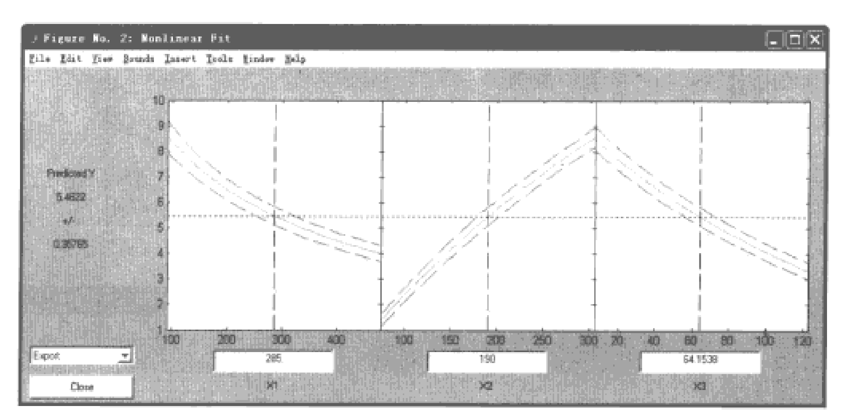


图 11.7　交互式多元非线性回归

窗口中显示了 X 的 3 个参量分别与 Y 轴的关系曲线，用鼠标可以移动某子图当前点的位置，另外两个子图的曲线进行相应移动。

11.5　小结

在科学研究中经常用到随机数，尤其是一些随机过程和现象的模拟。本章围绕统计工具箱介绍了在 MATLAB 中随机数的产生方法，包括不同分布的随机数的产生、随机排序、概率密度函数、累积概率值和逆累积分布函数。通过 Galton 板实验和赌徒输光问题的求解介绍了随机数的使用方法，同时介绍了关于随机数和离散数据统计分析方法，给出了均值、方差、协方差和相关系数等计算方法。最后介绍了回归分析，讨论了线性和非线性回归分析的求解方法，其在分析数据、建立经验公式和验证某一理论等方面都有重要应用。

◆　　◆　　◆

第 12 章 微分方程组的计算

本章包括

- **极限** 介绍利用函数 limit 求解极限问题。
- **全导数** 介绍函数 diff 和 jacobian 计算全导数。
- **dsolve 函数** 介绍利用函数 dsolve 求解微分方程（组）的解析解。
- **ode 系列函数** 详细介绍利用 MATLAB 函数计算微分方程组的数值解。
- **打靶法** 结合实例介绍打靶法的原理和求解过程实现。
- **时滞微分方程** 介绍时滞微分方程的求解方法。
- **偏微分方程** 主要介绍利用 PDE 工具箱求解偏微分方程组。
- **利用微分算积分** 介绍利用求解微分的程序来实现积分的计算。

微分是高等数学的基础内容，在很多问题模型中经常出现。在科学问题和工程研究中，极少数的问题可能得到解析解，而经常遇到函数表达式本身未知，需要通过实验测量数据或者进行数值方式的模拟。用户熟练掌握微分方程方面的内容对于求解一些问题具有重要帮助。本章介绍极限、导数以及微分方程（组）的解析和数值解法。结合 MATLAB 提供的一些函数来说明不同类型的微分方程（组）的解法。对于偏微分方程组，MATLAB 提供了 PDE 工具箱实现一些典型方程组的求解。

12.1 极限

在前面的章节介绍了可以用函数 limit 来计算极限，该函数的调用格式为：

```
L=limit(F)
L=limit(F,a)
L=limit(F,x,a)
L=limit(F,x,a,'left') 或者 L=limit(F,x,a,'right')
```

参数说明： L 是返回的极限值。F 是符号表达式。a 给出变量趋近的位置。x 用于指出对表达式中的变量进行求极限运算。left 和 right 分别表示对表达式求左极限和右极限。

这里给出一个利用函数 limit 求三角函数 sin(x)导数的例子。

```
syms x dx
df = limit([sin(x+dx)-sin(x)]/dx,dx,0)
```

输出结果为：

```
df = cos(x)
```

这个结果和 diff(sin(x))得到的结果是一致的。读者可以用函数 limit 来解决一些高等数学中的极限问题。

12.2 全导数

对于函数 $f(x,y,z)$ 全导数的计算需要得到下面 3 个导数值：$\frac{\partial f(x,y,z)}{\partial x}$，$\frac{\partial f(x,y,z)}{\partial y}$ 和 $\frac{\partial f(x,y,z)}{\partial z}$。利用 MATLAB 中的函数 diff 和 jacobian 可以得到这 3 个量的解析结果，其调用格式为：

```
D1=diff(F1,x);
D2=jacobian(F2,v);
```

参数说明： D1 和 D2 分别是函数 diff 和 jacobian 返回的导数值。F1 是函数表达式。F2 是由多个函数表达式组成的向量。x 用于表明对表达式 F1 中变量 x 求导数。v 是一个由变量组成的向量，jacobian 函数逐个对其中的变量计算导数表达式。

下面举例说明函数 diff 和 jacobian 的用法，利用它们计算 $f(x,y,z)=yz\sin(x)$ 的全导数分量。首先给出利用 diff 函数求解的程序：

```
syms x y z
f = y*z*sin(x);
dxf = diff(f,x)
dyf = diff(f,y)
dzf = diff(f,z)
```

输出如下：

```
dxf = y*z*cos(x)
dyf =z*sin(x)
dzf =y*sin(x)
```

用户还可以使用函数 jacobian 来计算全导数的 3 个分量，程序如下：

```
syms x y z
f = y*z*sin(x);
D = jacobian([f,f,f],[x,y,z]) % 把函数向量的分量写为同样函数，算出各个变量的导数，从而得到全导数
dxf = D(1,1)
dyf = D(2,2)
dzf = D(3,3)
```

输出为：

```
D = [ y*z*cos(x),    z*sin(x),    y*sin(x)]
[ y*z*cos(x),    z*sin(x),    y*sin(x)]
[ y*z*cos(x),    z*sin(x),    y*sin(x)]
dxf =y*z*cos(x)
dyf =z*sin(x)
dzf =y*sin(x)
```

可见函数 jacobian 不但计算出了全导数的 3 个分量，这 3 个分量位于矩阵 D 的主对角线上，而且其他的偏导数也给出来了。

12.3 dsolve 函数

函数 dsolve 可以用来求解一些微分方程（组）的解析解，其调用格式为：

```
S=dsolve('eqn1','eqn2', ...);
S=dsolve('eqn1','eqn2', ..., 'c1', 'c2', ...);
```

参数说明： S 为求解的输出结果。eqn1 和 eqn2 等表示微分方程。在微分方程中各阶微分项 $\frac{df}{dt}$，$\frac{d^2 f}{dt^2}$，$\frac{d^3 f}{dt^3}$ 和 $\frac{d^n f}{dt^n}$ 可以用下面的形式来表达：Df，D2f，D3f 和 Dnf。c1 和 c2 等是定解条件，即在某些特殊位置处的函数值或者导数值。

下面来举例说明函数 dsolve 的用法。求下面几个微分方程的解。

- $\frac{df}{dt} = f + 2t$
- $\frac{d^2 g}{dt^2} = g + \sin t$，定解条件为 $g(1)=2$，$g'(0)=1$
- $\frac{d^4 h}{dt^4} = t + \sin t$，定解条件为 $h(0)=1$，$h'(0)=2$，$h''(0)=3$，$h'''(0)=4$
- $\begin{cases} \frac{d^2 f_1}{dt^2} = t^2 \\ \frac{df_2}{dt} = t \end{cases}$，定解条件为 $f_1(0)=1$，$f_1'(0)=1$，$f_2(0)=2$

相关程序如下：

```
f = dsolve('Df=f+2*t')  % 求关于 f 的微分方程
g = dsolve('D2g=g+sin(t)','g(1)=2','Dg(0)=1')  % 求关于 g 的二阶微分方程
h = dsolve('D4h=t+sin(t)','h(0)=1','Dh(0)=2','D2h(0)=3','D3h(0)=4')  % 求关于 h 的
4 阶微分方程
[f1,f2] = dsolve('D2f1=t^2','Df2=t','f1(0)=1','Df1(0)=1','f2(0)=2')  % 求关于 f1
和 f2 的微分方程组
```

输出如下：

```
f =-2-2*t+exp(t)*C1
g
=1/2*exp(t)/(exp(1)+exp(-1))*(4+3*exp(-1)+sin(1))-1/2*exp(-t)*(3*exp(1)-4-sin(
1))/(exp(1)+exp(-1))-1/2*sin(t)
h =1/120*t^5+sin(t)+5/6*t^3+3/2*t^2+t+1
f1 =1/12*t^4+t+1
f2 =1/2*t^2+2
```

当没有定解条件时，函数 dsolve 会给出一个含有常数的结果。当有定解条件时，函数 dsolve 不能给出一个简单的解析结果，比如：

```
p = dsolve('Df=sin(f)*f')
q = dsolve('D2q=q^2')
```

执行上面两条语句后输出为：

```
p = RootOf(t-Int(1/sin(_a)/_a,_a =  .. _Z)+C1)
Warning: Explicit solution could not be found; implicit solution returned.
> In dsolve at 315
  In jacobian_test at 24
q =[    Int(3/(6*_a^3+3*C1)^(1/2),_a=..q)-t-C2=0,
Int(-3/(6*_a^3+3*C1)^(1/2),_a=..q)-t-C2=0]
```

可见 MATLAB 不能给出一个简单的结果，此时给出了一个相关的积分表达式。其中 int 表示求积分运算，而函数 RootOf 是一个特殊函数，用户利用 mhelp 可以查看它的信息。在使用 dsolve 求解一些稍微复杂一点的微分方程(组)的时候，结果可能会出现一些特殊函数，用户可以用 mhelp 来查看这些特殊函数的用法和含义。更复杂的微分方程需要考虑数值方法进行计算，这方面的内容将在后面介绍。

12.4 ode 系列函数

一些科学研究和工程设计中需要建立微分数学模型，一般情况下这些微分方程找不到合适的解析解，因此需要考虑数值方法求得问题的近似解。

对于微分方程的数值解法，MATLAB 提供了相关的求解函数，用户可以利用它们方便地求解微分方程（组）。首先给出 MATLAB 提供的求解微分方程组的函数及用法说明，如表 12.1 所示。

表 12.1 求解微分方程组的 MATLAB 函数

函数名	说明
ode45	高阶（4~5）的显式单步龙格-库塔法求解微分方程组，其可以求解中等精度要求的非刚性问题
ode23	低阶（2~3）的显式单步龙格-库塔法求解微分方程组，其可以求解弱刚性、低精度要求或者原函数不光滑（比如不连续）等问题
ode113	可变阶（1~13）的多步 Adams-Bsahforth-Moulton PECE 求解微分方程组。其可用于求解中等精度或者较高精度要求的非刚性问题，还包括原函数较难计算的情况。但是不适用于原函数不光滑的情况（即不连续或者低阶导数不连续）
ode15s	可变阶（1~5）的隐式多步法求解微分方程组，其适用于中等精度要求的刚性问题。在使用 ode45 函数求解失败或者计算速度低时，用户可以尝试使用这个函数
ode15i	可变阶（1~5）阶的隐式法求解微分方程组，其可以用来求解形如 $f(t,y,y')=0$ 的一类方程
ode23s	修正的隐式单步 Rosenbrock 二阶法求解微分方程组，其适用于低精度要求或者原函数不连续的刚性问题
ode23t	低阶的梯形法求解适当刚性的微分方程和微分代数方程
ode23tb	低阶的方法求解难度较大的微分方程



刚性问题是指基本时间常数按几个数量级变化的问题。

12.4.1 odeset 函数

在介绍这些函数用法之前，先介绍一下 odeset 函数的用法，这个函数可以设置这些微分方程

求解函数的一些参数。odeset 函数调用格式如下：

```
options = odeset('name1',value1,'name2',value2,...);
```

参数说明：options 是返回的一个结构体，其中包含了求解过程控制参数。name1 和 name2 是属性名。value1 和 value2 是对应的取值。下面把所有属性名及含义列成表格形式，如表 12.2 所示。

表 12.2 odeset 函数的属性名及含义

属性名	说明
RelTol	解的相对容许误差，每步估计误差满足 e(i)<=max(RelTol*abs(y(i)),AbsTol(i))。其默认值为 1e-3
AbsTol	解的绝对容许误差。当函数值接近于 0 时，存在控制误差。其默认值为 1e-6
NormControl	相对于解的范数控制误差，可取参数为 on 和 off。当取 on 时，误差满足条件较弱的标准，即 norm(e) <= max(RelTol*norm(y),AbsTol)
Refine	输出优化因子，其值为正整数。执行中按给定的因子用连续扩展公式对内部解进行插值来增加输出点数以产生更光滑的输出。例如当这个参数等于 4 时，每完成一步计算，产生 4 个等间距的输出点。当采样点数大于 2 时，这个参数不起作用
OutputFcn	每步计算后求解程序输出的函数。当用输出参数调用求解程序时，[t,y]=solver(...)，这里 solver 指求解微分方程的函数，此时 OutputFcn 是一个空的字符串。当无输出参数时，OutputFcn 的默认值为 odeplot
OutputSel	输出选择下标，传给 OutputFcn 解分量的下标。如 options=odeset('OutputSel',[1,3])的含义是把第 1 个和第 3 个解分量传递出来，作为输出
Stats	完成计算时，显示消耗的统计。可取参数为 on 和 off，默认值为 off
Jacobian	控制求解微分方程函数解析雅可比矩阵。可取参数为 on 和 off，默认值为 off。利用 odefile(t,y,' Jacobian')可得到雅可比矩阵
JPattern	控制 odefile 中得到雅可比矩阵的系数模式。可取参数为 on 和 off，默认值为 off。odefile(t,y,' JPattern')可得到含 1 的系数矩阵，用于显示雅可比矩阵的非零模式
Vectorized	快速计算数值雅可比矩阵的向量化 odefile。可取参数为 on 和 off，默认值为 off
Events	事件期望位置。可取参数为 on 和 off，默认值为 off。当这一项等于 on 时，odefile(t,y,' Events ')返回事件函数结果
Mass	从 odefile 中得到块矩阵。可取参数为 on 和 off，默认值为 off。当这一项等于 on 时，odefile(t,y,' Mass')返回块矩阵
MStateDependence	块矩阵的置信度，可取参数为 none、weak、strong，默认值是 weak
MassSingular	块矩阵是奇异矩阵，可取参数为 yes、no、maybe，默认值是 maybe
MvPattern	系数矩阵 dMv/dy
InitialSlope	初始斜率，其值满足关系(t0,y0)*yp0 = F(t0,y0)
InitialStep	初始步长，如果不合适将自动选择更好的步长
MaxStep	最大步长，默认值为变量整个采样区间长度的 1/10
NonNegative	非负解分量
BDF	在 ode15s 函数中采用向后差分公式，可取参数为 on 和 off，默认值为 off
MaxOrder	设置 ode15s 和 ode15i 函数的最大阶数，可取数值为 1，2，3，4，5，其中默认值是 5

下面开始介绍微分方程求解函数的使用。在表 12.1 中，除 ode15i 以外的所有 ode 系列函数的调用格式为：

```
[tout,yout] = solver(odefun,tspan,y0);
```

```
[tout,yout] = solver(odefun,tspan,y0,options);
[tout,yout] = solver(odefun,tspan,y0,options,p1,p2,…);
```

参数说明： tout 是输出的变量采样点系列。yout 是对应的函数的数值解。solver 指求解微分方程的 MATLAB 函数。odefun 是定义微分方程组的函数，它可以用函数 inline 或者一个函数文件来定义。tspan 是变量的求解区间。y0 是初始条件。optoins 是求解过程控制参数，其用 odeset 函数来定义。参数 p1，p2 等表示传递到定义微分方程的函数中进行计算。

12.4.2 函数 ode15i

函数 ode15i 可以用来求解一类形如 $f(t,y,y')=0$ 的微分方程，其调用格式如下：

```
[tout,yout] = ode15i(odefun,tspan,y0,yp0);
[tout,yout] = ode15i(odefun,tspan,y0,yp0,options);
[tout,yout] = ode15i(odefun,tspan,y0,yp0,options,p1,p2,...);
```

参数说明： 参数 yp0 是导数 y' 的初始条件，其他参数的含义与别的 ode 系列函数的调用形式相同。

12.4.3 举例说明

下面举例来说明这些函数的用法。

例 12-1： 求下面微分方程在区间[0,5]的解（初始条件为：$y(0)=1$，$y'(0)=0$）。

$y''+4y=3\sin 9t$

分析： ode45 函数不能直接求解这个方程，用户需要把这个方程转化为一阶方程组形式从而成为矩阵形式便于计算。本问题中的这个二阶微分方程可以等价地写为下面两个一阶微分方程组（初始条件为：$y_1(0)=1$，$y_2(0)=0$）。

$$\begin{cases} y_1' = y_2 \\ y_2' = 3\sin 9t - 4y_1 \end{cases}$$

其中 $y_1(t)$ 对应于函数 $y(t)$，而 $y_2(t)$ 对应于 $y'(t)$。（这里补充高阶微分方程的转化方法，对于 n 阶的微分方程用户可以考虑类似地来转化。用 $y_1(t)$，$y_2(t)$，$y_3(t)$，…，$y_n(t)$ 来代替 $y(t)$，$y'(t)$，$y''(t)$，…，$y^{(n-1)}(t)$，前 $n-1$ 个方程是 $y_1'=y_2$，$y_2'=y_3$，$y_3'=y_4$，…，$y_{n-1}'=y_n$，最后一个方程是用 $y_1(t)$，$y_2(t)$，$y_3(t)$，… $y_n(t)$ 来替换高阶微分方程中的各阶导数，从而得到 n 个一阶微分方程组。）然后用户可建立函数文件来定义这个微分方程组或者使用函数 inline 来定义一个函数。下面分别介绍这两种方式来定义微分方程组。

首先介绍使用函数文件定义微分方程组的方法，相关程序如下：

```
function dy = tfys(t,y);
dy(1,1) = y(2);                    % 对应于方程组中的第一个微分方程
dy(2,1) = 3*sin(9*t)-4*y(1);       % 对应于方程组中的第二个微分方程
```

上面的函数文件 tfys.m 用来定义微分方程组，其中输出结果 dy 要求必须是列向量。接下来就可以调用函数 ode45 来求解微分方程了，程序如下：

```
tspan = [0,5];        %定义变量求解区间
y0 = [1,0];           % 定义初值
```

```
[t,y] = ode45(@tfys,tspan,y0);  % 调用函数 ode45 求解方程
figure;
plot(t,y(:,1),'k-');            % 画出函数 y(t)的曲线
hold on;
plot(t,y(:,2),'k:');            % 画出函数 y(t)的导数曲线
set(gca,'Fontsize',12);         % 重置坐标轴字体大小
xlabel('\itt','Fontsize',16);   % 标注 X 轴含义
```

下面给出使用函数 inline 结合函数 ode45 的程序代码：

```
tspan = [0,5];                                      % 定义变量求解区间
y0 = [1,0];                                         % 定义初值
fun = inline('[y(2);3*sin(9*t)-4*y(1)]','t','y');   % 定义微分方程组
[t,y] = ode45(fun,tspan,y0);                        % 调用函数 ode45 求解方程
% 此处略去绘图程序，用户可以复制上面的语句来绘图
```

上面使用函数 inline 定义微分方程的程序得到如图 12.1 所示的图形。

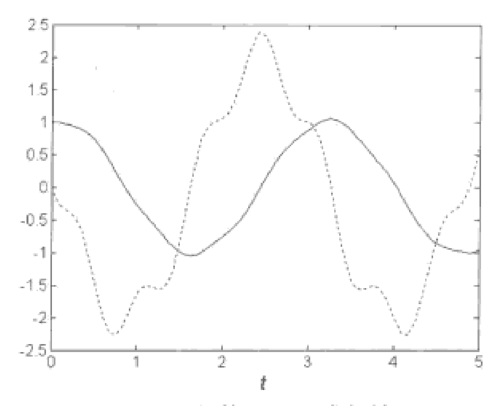


图 12.1 函数 ode45 求解结果

例 12-2：在区间[0,14]求解下面含系数的微分方程组。

$$\begin{cases} x_1' = ax_1 - bx_1x_2 \\ x_2' = cx_1x_2 - dx_2 \end{cases}$$

其中 $a = 2$ ，$b = 0.01$ ，$c = 0.001$ ，$d = 0.7$ 。初始值为 $x_1(0) = 300$ ，$x_2(0) = 100$ 。

分析：在这个问题中，直接是一阶微分方程组成的方程组，用户不必再进行转化了。不同的是这里含有 4 个不同的系数参数，并给出一种传递系数参数的方法，同时利用函数文件和函数 inline 来定义微分方程组。求解微分方程的函数仍然使用 ode45。首先给出使用函数的文件来定义这个微分方程组，具体程序如下：

```
function dx=preyer(t,x,flag,a,b,c,d);
dx(1,1) = a*x(1)-b*x(1)*x(2);       % 定义微分方程组中第一个方程
dx(2,1) = c*x(1)*x(2)-d*x(2);       % 定义微分方程组中第二个方程
```

在这个例子中，要传递的参数按顺序依次写在输入参数 flag（用户也可以写入其他名称，但不能和变量名重复）后面。这里 flag 只是一个标记，在函数的计算中没用到它。用下面的程序就可以来求解微分方程了。

```
tspan = [0,14];         % 定义变量求解区间
```

```
x0 = [300,100];       % 定义初值
a = 2;                % 参数赋值
b = 0.01;             % 参数赋值
c = 0.001;            % 参数赋值
d = 0.7;              % 参数赋值
options = odeset('RelTol',1e-6);                  % 设置相对误差
[t,x] = ode45('preyer',tspan,x0,options,a,b,c,d);    % 调用函数 ode45 求解方程
figure;
plot(t,x(:,1),'k-');              % 画出函数 x1(t)的曲线
hold on;
plot(t,x(:,2),'k:');              % 画出函数 x2(t)的曲线
set(gca,'Fontsize',12);         % 重置坐标轴字体大小
xlabel('\itt','Fontsize',16);      % 标注 X 轴含义
L=legend('{\itx}_1','{\itx}_2',0);       % 标注线性对应的函数值
set(L,'Fontname','Times New Roman');      % 设置 legend 标注文字的字体名
set(gcf,'color','w','Position',[121 233 1121 406]);    % 重置图形窗口的背景色、位置和
大小
```

上述程序输出结果如图 12.2 所示。

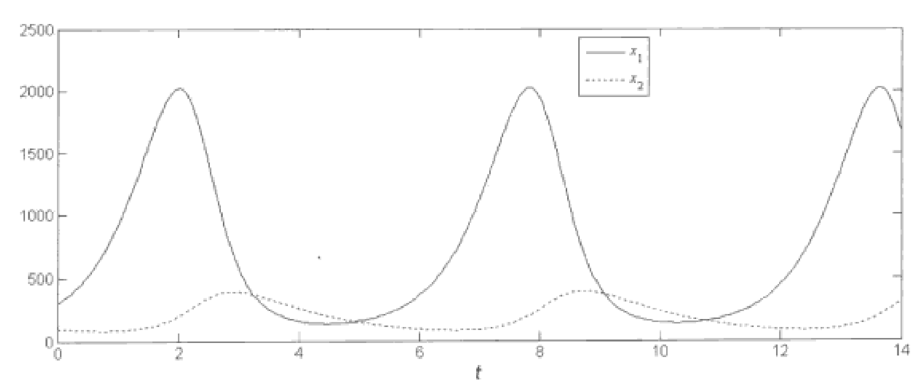


图 12.2　ode45 函数求解微分方程组的结果

如果把上面程序中的“options = odeset('RelTol',1e-6);”改为语句“options = odeset('RelTol',1e-3);”，结果如图 12.3 所示。

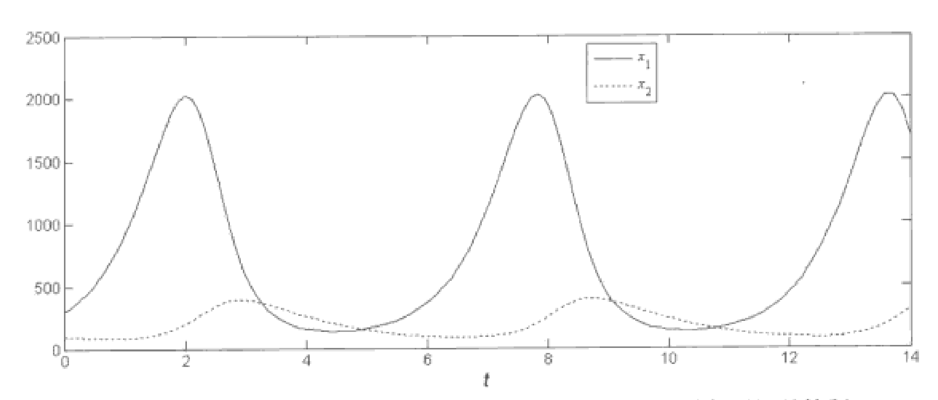


图 12.3　ode45 函数求解微分方程组的结果，使用不同的相对误差控制

对比图 12.2 和图 12.3 中实曲线的第 3 个峰可以发现：相对误差 RelTol 较大时，结果有一个平顶。因此需要使用相对误差小的结果来求解这个问题。微分方程可以采用下面的代码：

```
tspan = [0,14];        % 定义变量求解区间
x0 = [300,100];       % 定义初值
a = 2;               % 参数赋值
b = 0.01;            % 参数赋值
c = 0.001;           % 参数赋值
d = 0.7;             % 参数赋值
options = odeset('RelTol',1e-6); % 设置相对误差
funs=inline('[a*x(1)-b*x(1)*x(2);c*x(1)*x(2)-d*x(2)]','t','x','flag','a','b','c','d');
[t,x] = ode45(funs,tspan,x0,options,a,b,c,d); % 调用函数 ode45 求解方程
```

上面这段程序所得结果和用函数文件定义的微分方程组的结果一样。在例 12-2 中，workspace 里面的变量可以传递到微分方程组的计算程序中。这样的形式用户可以再使用其他程序来调用这个求解微分方程的程序，更换微分方程中的系数来比较不同参数对于结果的影响。

从例 12-1 和例 12-2 中可以发现：利用函数 inline 定义的代码可以使程序代码简化，并且可以在同一个文件中完成微分方程的求解，便于文件的保存。但是利用函数 inline 会使程序的执行速度降低。利用函数文件定义的微分方程程序不能作为一个子函数和 ode45 函数文件放在一起，函数文件定义微分方程组的好处是：可以用来定义比较复杂的微分方程，同时计算速度较函数 inline 定义的方式快。



利用引号标识对于不同情况都是适用的，而利用符号@标识只能用于函数文件定义的微分方程且没有参数传递的情况。对于初次使用 ode 系列函数的用户，建议使用引号标识 odefun。

例 12–3：用 ode23 和 ode23s 解微分方程组。

函数 ode23 和 ode23s 可以分别用来求解弱刚性和不连续刚性的微分方程，它们的调用格式为以 12.4.1 节介绍的统一格式。下面具体来说明它们的用法。

$$\begin{cases} y_1' = a-(b+1)y_1 + y_1^2 y_2 \\ y_2' = by_1 - y_1^2 y_2 \end{cases}$$，其中 $a=500$，$b=2$。初始条件为：$y_1(0)=3$，$y_2(0)=4$。

分析：在这个问题中含有变化的参数 a 和 b 仍然需要进行参数传递。同时这里考虑使用函数 ode23 和 ode23s 来计算这个微分方程，并进行求解方程时间的比较。首先给出定义微分方程组的函数文件内容：

```
function dy = adans(t,y,flag,a,b);
dy(1,1) = a-(b+1)*y(1)+y(1).^2*y(2);  % 对应第一个微分方程
dy(2,1) = b*y(1)-y(1).^2*y(2);        % 对应第二个微分方程
```

在下面调用 ode23 求解这个微分方程组的程序中，加入了函数 cputime 计算花费的时间。

```
tspan = [0,10];        % 定义变量求解区间
y0 = [3,4];           % 定义初值
a = 500;             % 参数赋值
b = 2;               % 参数赋值
options = odeset('RelTol',1e-3);  % 设置相对误差
t0 = cputime;
[t,y] = ode23('adans',tspan,y0,options,a,b);  % 调用函数 ode23 求解方程
time = cputime-t0
```

```
figure;
plot(t,y(:,1),'k-');          % 画出函数y1(t)的曲线
hold on;
plot(t,y(:,2),'k:');          % 画出函数y2(t)的曲线
set(gca,'Fontsize',12);     % 重置坐标轴字体大小
ylim([0,600]);            % 重置Y轴范围
xlabel('\itt','Fontsize',16);  % 标注X轴含义
set(gcf,'color','w');
```

上述程序中调用 ode23 函数花费的时间（单位：秒）是：

```
time =  392.4375
```

上面的计算时间会因计算机条件不同而异，对于同一台计算机也会因为调用应用程序的情况不同而不同。这个例子中，使用 ode45 函数计算也会花费较长的时间。如果把函数 ode23 换为 ode23s，所求得的微分方程的结果相同，但是花费的时间是：

```
time =    0.6250
```

此外，用户如果利用函数 ode15s，ode23t 和 ode23tb 也可以较快地得到计算结果，其花费时间和函数 ode23s 相差不多。可见当计算速度较慢的时候，用户可以考虑更换求解的 MATLAB 函数。利用函数 ode23 求解花费的时间会随着 b/a 的比值增大而快速增加。从图 12.4 中可以看出结果变化很快，因此这个问题属于刚性问题，应该使用那些适合求解刚性微分方程的 MATLAB 函数。对于不同的问题，用户需要根据实际情况来选用求解函数。

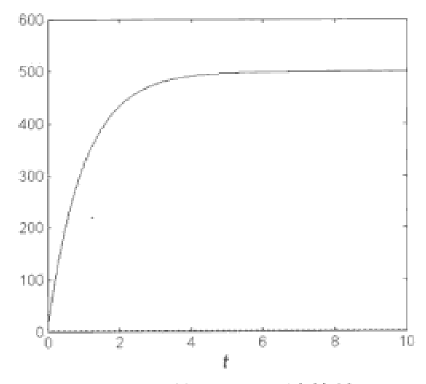


图 12.4　利用 ode23 计算结果

例 12-4：用 ode23tb 函数求下面的微分方程组在区间[0,20]上的解。

函数 ode23tb 使用低阶方法求解难度较大的微分方程，一些特殊的微分方程可以尝试使用这个函数求解。下面给出相关的例子。

$\begin{cases} x_1' = x_2 - f(t) \\ x_2' = x_1 g(t) - x_2 \end{cases}$，其中 $f(t) = \begin{cases} \sin(t), & t < 4\pi \\ 0, & t \ge 4\pi \end{cases}$，$g(t) = \begin{cases} 0, & t < 7\pi/2 \\ 2\cos t, & t \ge 7\pi/2 \end{cases}$。初始条件是：$x_1(0)=1$，$x_2(0)=2$。

分析：这个问题与前面的不同在于微分方程中有两个分段函数，可以在函数文件中计算这两个分段函数，用户可结合前面介绍的分段函数的定义方法。

这里先给出定义微分方程组的程序：

```
function dx = odeppiece(t,x);
ft = 0;            % 代表函数 f(t)
gt = 0;            % 代表函数 g(t)
if t<4*pi
   ft = 2*sin(t);
end
if t>pi*3.5
   gt = cos(t);
end
dx(1,1) = x(2)-ft;        % 定义第一个微分方程
dx(2,1) = x(1)*gt-x(2);   % 定义第二个微分方程
```

上面程序中，if 语句用来定义两个分段函数。下面调用 ode23t 函数来计算这个微分方程：

```
tspan = [0,20];       % 定义变量求解区间
x0 = [0,1];          % 定义初值
[t,x] = ode23tb('odeppiece',tspan,x0); % 调用函数 ode23tb 求解方程
figure;
plot(t,x(:,1),'k-');              % 画出函数 x1(t)的曲线
hold on;
plot(t,x(:,2),'k:');              % 画出函数 x2(t)的曲线
set(gca,'Fontsize',12);          % 重置坐标轴字体大小
L=legend('{\itx}_1','{\itx}_2',0);
set(L,'Fontname','Times New Roman')
xlabel('\itt','Fontsize',16);     % 标注 X 轴含义
set(gcf,'color','w');
```

所得结果如图 12.5 所示。无论求解微分方程如何复杂，用户都可以根据关系在函数文件中建立微分方程的模型。

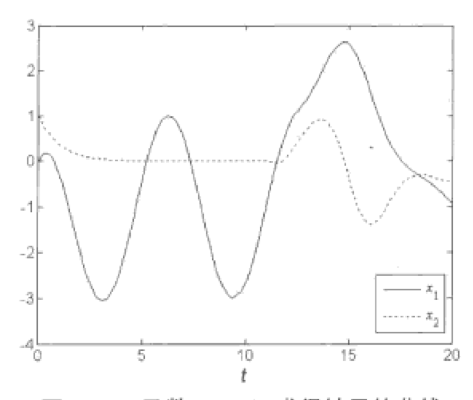


图 12.5　函数 ode23t 求得结果的曲线

函数 ode15i 可以用来求解隐式格式的微分方程，如 $f(t,x,x')=0$。下面给出一个隐式微分方程的例子。

例 12-5： 求解下面微分方程的数值解。

$$\begin{cases} x_2'\left[x_2\cos(4t)-x_1^3\right]-tx_1'x_2'=0 \\ t\sin x_2/8-2x_2x_2'=0 \end{cases}$$

初始条件为 $x_1(0)=1$，$x_2(0)=1$，$x_1'(0)=-0.2$，$x_2'(0)=0.4$。

先定义这个隐式的微分方程组，其与前面介绍的定义微分方程的方法不同。具体程序内容如下：

```
function D = impfun(t,x,xp);
D = [xp(2)*[cos(t*4)*x(2)-x(1)^3]-xp(2)*xp(1)*t;...
   t*sin(x(2))/8-xp(2)*x(2)*2];        % 定义隐式微分方程组
```

上述微分方程组的定义是按照 $D=f(t,x,x')$ 格式输入的，把每个等式的左侧内容按语法格式输入。函数文件中的 x 和 xp 分别为原函数分量组成的向量和原函数的导数分量组成的向量。如果存在需要传递的系数，可以按前面的方式来定义，即 "D = impfun(t,x,xp,flag,p1,p2,...);"。

然后调用 ode15i 函数就可以来求解这个隐式微分方程组了。

```
tspan = [0,30];              % 定义变量求解区间
x0 = [1,1]';                 % 定义初值
xp0 = [-0.2,0.4]';           % 定义初值
[t,x] = ode15i('impfun',tspan,x0,xp0);
      % 调用函数 ode15i 求解方程
figure;
plot(t,x(:,1),'k-');            % 画出函数 x1(t)的曲线
hold on;
plot(t,x(:,2),'k:');            % 画出函数 x2(t)的导数曲线
set(gca,'Fontsize',12);        % 重置坐标轴字体大小
L=legend('{\itx}_1','{\itx}_2',0);
set(L,'Fontname','Times New Roman')
xlabel('\itt','Fontsize',16);     % 标注 X 轴含义
set(gcf,'color','w');
```

上述程序所得结果如图 12.6 所示。

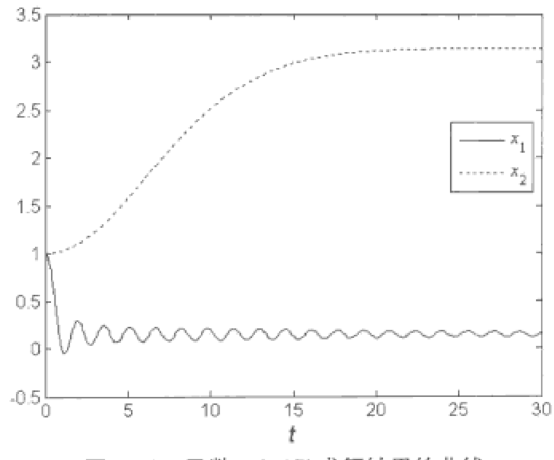


图 12.6　函数 ode15i 求解结果的曲线

下面介绍微分代数方程（differential algebraic equation，缩写为 DAE）的解法。在这类方程中，一些变量要受到代数方程的约束。在这类方程组中同时含有微分方程和代数方程，这时用户不能直接利用前面介绍的 ode 系列函数求解。一般地，微分代数方程可以写为下面的形式：

$$M(t,x)x'=f(t,x) \tag{12-1}$$

其中 x 是由自变量组成的列向量 $M(t,x)$，是一个奇异矩阵，这一项对应函数 odeset 属性里面的变量 mass。利用这个参数就可以求解这类方程了。

例 12-6：求解下面的微分方程的数值解。

$$\begin{cases} x_1' = -0.3x_1 - 2x_2x_3 + x_2x_4 \\ x_2' = -x_2 + 0.5x_4 - 0.2\sin 0.6t \\ x_3' = -0.2x_2x_1 + x_3x_4 \\ 1 = x_3 + x_4 - x_1 - x_2 \end{cases}$$，初值条件是 $x_1(0) = -1$， $x_2(0) = 1$， $x_3(0) = 1$， $x_4(0) = 0$。

分析：其中最后一个方程是一个代数方程，来约束 4 个分量之间的关系，而前 3 个方程是典型的一阶微分方程。这个问题的方程可以转为下面的形式：

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1' \\ x_2' \\ x_3' \\ x_4' \end{bmatrix} = \begin{bmatrix} -0.3x_1 - 2x_2x_3 + x_2x_4 \\ -x_2 + 0.5x_4 - 0.2\sin 0.6t \\ -0.2x_2x_1 + x_3x_4 \\ x_1 + x_2 - x_3 - x_4 + 1 \end{bmatrix} \tag{12-2}$$

这个表达式和式（12-1）是一致的，因此可把前面的系数矩阵放在 odeset 的 Mass 属性里。根据上面的分析来建立定义微分方程的程序以及调用 ode 系列函数的主程序。首先给出定义微分方程的函数文件程序，内容如下：

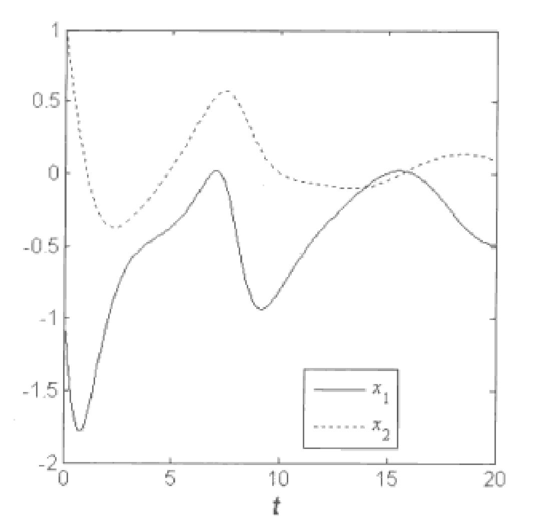
```
function dx = daefun(t,x);
dx = [-0.3*x(1)-2*x(2)*x(3)+x(2)*x(4);...
    -x(2)+0.5*x(4)-sin(t*0.6)*0.2;...
    -x(2)*x(1)*0.2+x(3)*x(4);...
    x(1)+x(2)-x(3)-x(4)+1];
```

上述这段程序对应着式（12-2）右侧的部分，与前面定义微分方程组的格式一样。

在定义完上述函数文件之后，用户就可以调用 ode15s 函数来计算这个微分代数方程了，具体程序内容如下：

```
tspan = [0,20];                              % 定义变量求解区间
x0 = [-1,1,1,0]';                             % 定义初值
M=eye(4);
M(4,4)=0;                                   % 定义系数矩阵 M(t,y)
options = odeset('Mass',M);
[t,x] = ode15s(@daefun,tspan,x0,options);    % 调用函数 ode15s 求解方程
figure;
subplot(121);
plot(t,x(:,1),'k-');                          % 画出函数 x1(t)的曲线
hold on;
plot(t,x(:,2),'k:');                          % 画出函数 x2(t)的导数曲线
set(gca,'Fontsize',12);                      % 重置坐标轴字体大小
L=legend('{\itx}_1','{\itx}_2',0);
set(L,'Fontname','Times New Roman')
xlabel('\itt','Fontsize',16);                   % 标注 X 轴含义
subplot(122);
plot(t,x(:,3),'k-');                          % 画出函数 x3(t)的曲线
hold on;
plot(t,x(:,4),'k:');                          % 画出函数 x4(t)的导数曲线
set(gca,'Fontsize',12);                      % 重置坐标轴字体大小
L=legend('{\itx}_3','{\itx}_4',0);
set(L,'Fontname','Times New Roman')
xlabel('\itt','Fontsize',16);                   % 标注 X 轴含义
set(gcf,'color','w');
```

上述程序执行之后得到的结果如图 12.7 所示。



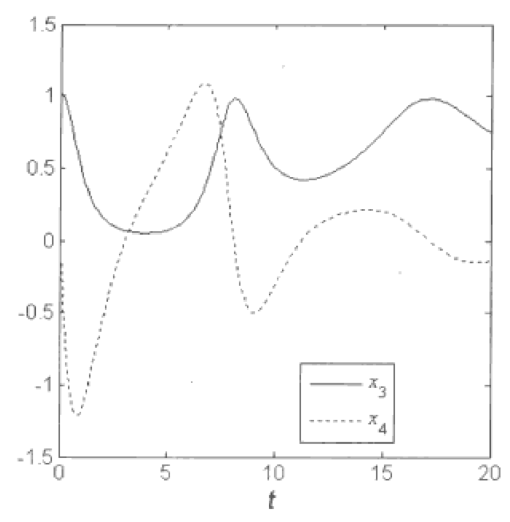


图 12.7 用函数 ode15s 求解微分代数方程的结果

这里需要补充两点：一是这里可以使用函数 ode15s 和 ode23t 得到正确的结果，其他 ode 系列函数，如 ode23s，ode23tb，ode45，ode23 和 ode113 不能得到正确结果；二是 odefun 的表示方法需要使用符号@，而用户如果把语句“[t,x] = ode15s(@daefun,tspan,x0,options);”改为“[t,x] = ode15s('daefun',tspan,x0,options);”，则会弹出如下错误提示：

```
??? SWITCH expression must be a scalar or string constant.
Error in ==> odemass at 98
  switch(mass)
Error in ==> ode15s at 267
[Mtype, Mt, Mfun, Margs, dMoptions] = odemass(FcnHandlesUsed,odeFcn,t0,y0,...
Error in ==> exampe12_6 at 7
[t,x] = ode15s('daefun',tspan,x0,options);   % 调用函数 ode15s 求解方程
```

例 12-7：用 ode15i 求解下面隐式微分代数方程。

$$\begin{cases}-0.39x_2' + \cos x_2^2 \sin 0.1x_3' = 0 \\ x_3' \cos t + x_3 \cos x_3 + \cos 6t = 1 \\ x_1 + x_2 = 1\end{cases}$$

分析：这里把这个微分方程转化为下面的形式：

$$\begin{cases}-0.39x_2' + \cos x_2^2 \sin 0.1x_3' = 0 \\ x_3' \cos t + x_3 \cos x_3 + \cos 6t = 1\end{cases}$$

此时上面的方程是一个隐式微分方程形式，而分量 x_1 可以用 $1-x_2$ 来计算。为了编程方便，建立如下的映射关系：$y_1 \to x_2$ 和 $y_2 \to x_3$。对应的方程为：

$$\begin{cases}-0.39y_1' + \cos y_1^2 \sin 0.1y_2' = 0 \\ -y_2' \cos t + y_2 \cos y_2 + \cos 6t = 1\end{cases}$$

首先给出定义这个隐式微分方程的函数文件内容：

```
function D = impdaef(t,y,yp);
D  = [-yp(1)*0.39+cos(y(1)^2)*sin(0.1*yp(2));...
```

```
-yp(2)*cos(t)+y(2)*cos(y(2))+cos(t*6)-1];
```

使用下面的程序调用 ode15s 函数来计算这个隐式微分方程：

```
tspan = [0,10];          % 定义变量求解区间
y0 = [0.5,0.2]';          % 定义初值
yp0 = [0,0]';
options = odeset('RelTol',1e-5);
[t,y] = ode15i('impdaef',tspan,y0,yp0,options);
figure; % 调用函数 ode15i 求解方程
plot(t,y(:,1),'k-');           % 画出函数 x2(t)的曲线
hold on;
plot(t,y(:,2),'k:');           % 画出函数 x3(t)的导数曲线
set(gca,'Fontsize',12);    % 重置坐标轴字体大小
L=legend('{\itx}_2','{\itx}_3',0);
set(L,'Fontname','Times New Roman')
xlabel('\itt','Fontsize',16);   % 标注 X 轴含义
set(gcf,'color','w');
```

上述程序所得结果如图 12.8 所示，其中 x_1 分量的结果没有在这里显示，可以通过简单的代数关系得到。从这个例子中可以发现求解微分代数方程的另一个思路就是：把其中的代数方程转化为另外一种形式，即用其他分量来表示其中的一个分量，如 $x_k = g\left(x_1, x_2, \ldots, x_{k-1}, x_{k+1}, \ldots, x_n\right)$，再把这个结果带入其他微分方程进行消元处理，从而可能得到一个隐式微分方程组，继而调用函数 ode15i 来计算。

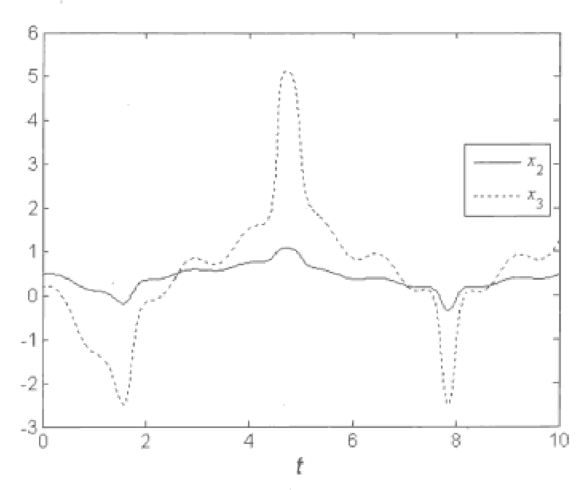


图 12.8 隐式微分代数方程求解结果

除了上面的例子，用户还可以指定采样点的位置，即 tspan 是很多个采样点组成的向量。下面举例说明这种情况的解法。

例 12–8：求下面两个微分方程在区间 $[0,10]$ 内的解。

$$\begin{cases} x_1' = -0.2x_2 - \sin 2t \\ x_2' = -0.1x_1 + 3\cos t \end{cases}$$，初值条件为为 $x_1(0)=2$，$x_2(0)=0$。

$y'' - 0.1(y')^4 - 0.3yt\cos 5t = 0$，定解条件为 $y(10)=0$，$y'(10)=-0.2$。

分析：第一个方程是一个标准的初值问题，使用函数 ode45 即可求解。在第二个问题中，给

定的定解条件是在终点位置的函数值和导数值，这里将引入负步长的概念求解，这样就可以使用 ode45 函数求解。相应的程序如下：

```
fun1 =  inline('[-x(2)*0.2-sin(2*t);-x(1)*0.1+3*cos(t)]','t','x'); % 定义第一个微分方程组
tspan1 = linspace(0,10,400);      % 生成采样点
x0 = [2,0]';                      % 设置初值
[tx,x] = ode45(fun1,tspan1,x0);   % 调用 ode45 函数求解
fun2 = inline('[y(2);y(2)^4*0.1+0.3*y(1)*t*cos(5*t)]','t','y');  % 定义第二个微分方程组
tspan2 = fliplr(tspan1);          % 生成负步长采样点
y0 = [0,-0.2];                    % 设置初值
[ty,y] = ode45(fun2,tspan2,y0);   % 调用 ode45 函数求解
figure;
subplot(121);
plot(tx,x(:,1),'k');              % 画出 x1(t)对应的曲线
hold on;
plot(tx,x(:,2),'k:');             % 画出 x2(t)对应的曲线
set(gca,'Fontsize',12);           % 重置坐标轴字体大小
L1 = legend('{\itx}_1','{\itx}_2',0);
xlabel('\itt','Fontsize',16);     % 标注 X 轴含义
subplot(122);
plot(ty,y(:,1),'k');              % 画出 y(t)对应的曲线
set(gca,'Fontsize',12);           % 重置坐标轴字体大小
L2 = legend('{\ity}',0);
set([L1,L2],'Fontname','Times New Roman');   % 设置标注字体
xlabel('\itt','Fontsize',16);                % 标注 X 轴含义
set(gcf,'color','w');
```

上面程序计算结果如图 12.9 所示。

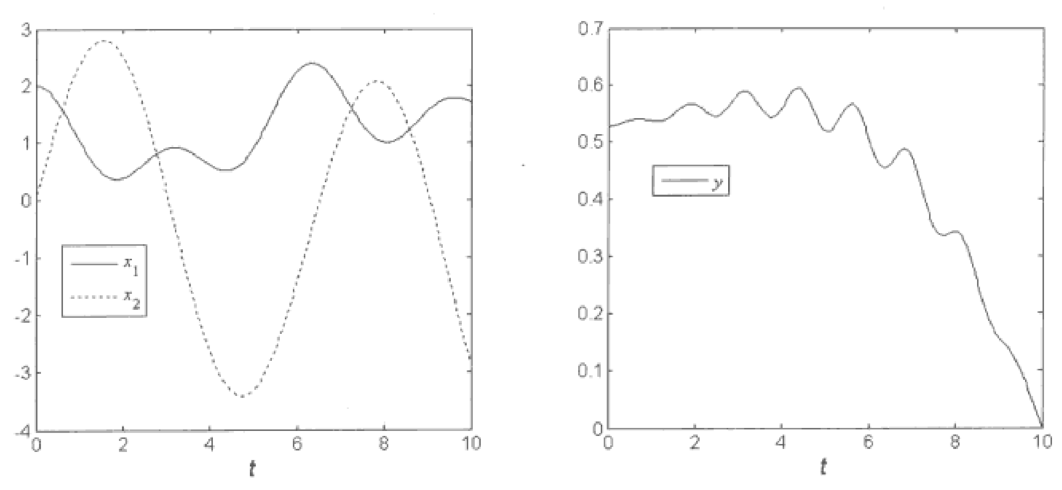


图 12.9　设置采样点的求解结果

输出的变量点 tx 和 ty 分别等于 tspan1 和 tspan2。对于 ode 系列函数，当 tspan 含 2 个元素的时候，tspan 的第 2 个元素值也可以小于第 1 个元素，程序正常执行。

在实际应用中，用户需要把微分方程进行一定的变形，使之适合 MATLAB 来求解。大多数带有初值条件的微分方程使用 ode 系列函数是可以求解的。同时微分方程可能受到求解步长、初值

以及方程中参数的影响而结果变化很大，有时用户需要结合自己的专业知识来检查结果的物理意义正确与否。

12.5 打靶法

前面介绍的 ode 系列函数只能直接用来求解初值问题，而不能直接用于求解边值问题。有时会遇到这样一类情况，即知道开始时刻和结束时刻的函数值，比如热传导问题，在初始时热源情况已知，一定时间后温度达到均匀分布；再如弦振动问题，端点的位置是固定的。这类问题被称为边值问题，可以使用如下方程来描述：

$$f(t,x,x')=0, \tag{12-3}$$

其中定解条件为：从 $x(0)=a$，$x(t_0)=b$，$x'(0)=c$，$x'(t_0)=d$ 四式中的两个端点 $t=0$ 和 $t=t_0$ 处各取出一个式子而得到端点的两个式子。$[0,t_0]$ 为求解区间，a，b，c 和 d 为已知数。下面将介绍求解这类方程的打靶法。

打靶法的原理如图 12.10 所示。假设枪和靶子分别放在 $t=0$ 和 $t=t_0$，而在 $t=0$ 平面上枪的位置和俯仰角可以分别认为是函数值和其导数值。对于边值问题，在 $t=0$ 处，函数值或者导数值是已知的，也就是说枪的位置或者俯仰角是已知的。持枪者通过调整枪的位置或者俯仰角就可以打中目标了。因此把这种思路称为打靶法。

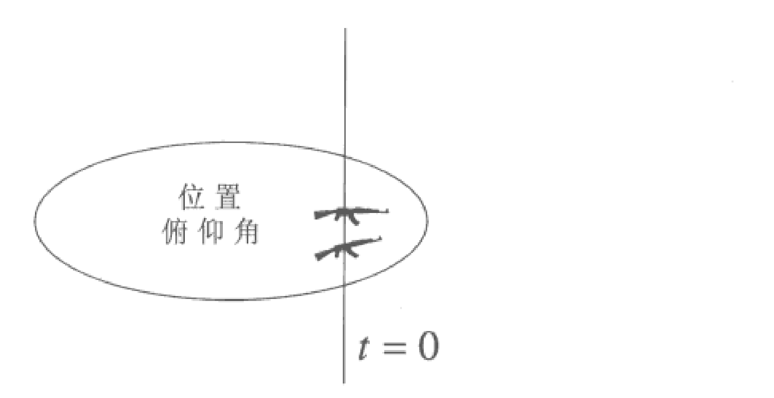


图 12.10 打靶法原理

线性边值问题可以用下面的公式来描述：

$$y''=p(t)y'(t)+q(t)y+r(t) \tag{12-4}$$

$y(a)=A$，$y(b)=B$ 在区间 $[a,b]$ 上有唯一解 $y=y(t)$。

求解步骤为：

step 1 求出下面的齐次微分方程在区间 $[a,b]$ 上的解并得出 $y_1(b)$ 的值：
$y_1''=p(t)y_1'+q(t)y_1$，初值条件为 $y_1(a)=1$，$y_1'(a)=0$。

step 2 求出下面的齐次微分方程在区间 $[a,b]$ 上的解并得出 $y_2(b)$ 的值：
$y_2''=p(t)y_2'+q(t)y_2$，初值条件为 $y_2(a)=0$，$y_2'(a)=1$。

step 3 求出下面的微分方程在区间 $[a,b]$ 上的解并得出 $y_3(b)$ 的值：

$y_3''=p(t)y_3'+q(t)y_3+r(t)$，初值条件为 $y_3(a)=0$， $y_3'(a)=0$。

step 4 计算参数 m 的值（ m 是初值中的导数值），其表达式如下：

$$m=\frac{B-Ay_1(b)-y_3(b)}{y_2(b)}$$



若 $y_2(b)=0$，方程无唯一解。

原方程组的解表示为： $y''=p(t)y'(t)+q(t)y+r(t)$，初值条件为 $y(a)=A$， $y'(a)=m$。

根据上面的计算步骤，可以编写出线性打靶法的计算程序，如下：

```
function [t,y] = lineshoot(f1,f2,tspan,x0f,varargin);
% 线性打靶法求解程序
[t,y1] = ode45(f2,tspan,[1,0] ,varargin);     % 计算函数y1(t)
[t,y2] = ode45(f2,tspan,[0,1] ,varargin);     % 计算函数y2(t)
[t,y3] = ode45(f1,tspan,[0,0] ,varargin);     % 计算函数y3(t)
m = [x0f(2)-x0f(1)*y1(end,1)-y3(end,1)]/y2(end,1);   % 求参数m
[t,y] = ode45(f1,tspan,[x0f(1),m] ,varargin);          %求出原微分方程的解
```

参数说明：输出 t 为变量离散数据。输出 y 为函数分量的离散数据。f1 为齐次微分方程的函数。f2 为原微分方程对应的函数。tspan 为求解区间。x0f 为边值条件。其中，4 次调用 ode45 函数求解方程。参数 varargin 对应于微分方程其他输入参数，如 options 和传递到微分方程中的参数。

例 12-9：求下面边值问题在区间[0,4]上的解，初值条件为： $y(0)=1$， $y(4)=2$。

$y''=2y'\cos t-y\sin 4t-\cos 3t$

分析：这是一个边值问题，其中系数函数 $p(t)=2\cos t$， $q(t)=-\sin 4t$， $r(t)=-\cos 3t$。利用前面建立的函数文件 lineshoot.m 可以求解这个微分方程的解。在求解的时候需要把这个二阶微分方程转化为一阶微分方程组。方便起见，这里使用函数 inline 来定义微分方程组。求解程序如下：

```
f1 =inline('[y(2);2*cos(t)*y(2)-sin(4*t)*y(1)-cos(3*t)]','t','y');  % 定义原微分
方程
f2 =inline('[y(2);2*cos(t)*y(2)-sin(4*t)*y(1)]','t','y');         % 定义其次微分方程
tspan = [0,4];                   % 求解区间
x0f=[1,2];                      % 边值条件
[t,y] = lineshoot(f1,f2,tspan,x0f); % 调用线性打靶法求解函数 lineshoot
figure;
plot(t,y(:,1),t,y(:,2),'k:');
set(gcf,'color','w');
```

上述程序所得结果如图 12.11 所示，其中实曲线表示函数 $y(t)$，虚曲线对应其导数。

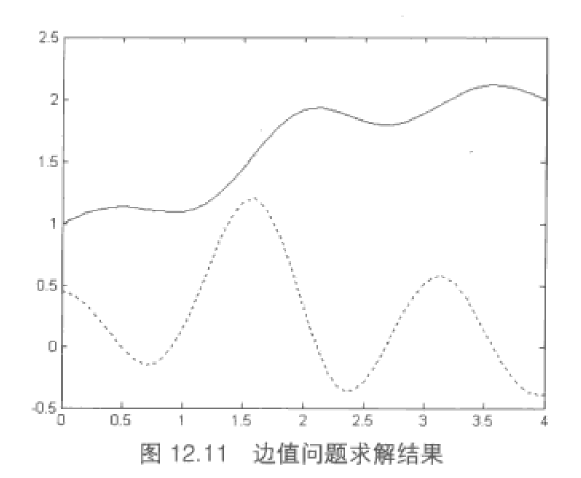


图 12.11 边值问题求解结果

下面来研究另一类线性边值问题，其可以用如下公式来描述：

$$y''=p(t)y'(t)+q(t)y+r(t) \tag{12-5}$$

$y'(a)=A$，$y'(b)=B$ 在区间$[a,b]$上有唯一解 $y=y(t)$。求解的 4 个步骤为：

step 1 求出下面的齐次微分方程在区间$[a,b]$上的解并得出 $y_1'(b)$ 的值：
$y_1''=p(t)y_1'+q(t)y_1$，初值条件为 $y_1(a)=1$，$y_1'(a)=0$。

step 2 求出下面的齐次微分方程在区间$[a,b]$的解并得出 $y_2'(b)$ 的值：
$y_2''=p(t)y_2'+q(t)y_2$，初值条件为 $y_2(a)=0$，$y_2'(a)=1$。

step 3 求出下面的微分方程在区间$[a,b]$上的解并得出 $y_3'(b)$ 的值：
$y_3''=p(t)y_3'+q(t)y_3+r(t)$，初值条件为 $y_3(a)=0$，$y_3'(a)=0$。

step 4 计算出参数M 的值（M 是初值中的导数值），其表达式如下：

$$M=\frac{B-Ay_2'(b)-y_3'(b)}{y_1'(b)}$$



若 $y_1'(b)=0$，方程无唯一解。

原方程组的解表示为：$y''=p(t)y'(t)+q(t)y+r(t)$，初值条件为 $y(a)=A$，$y'(a)=M$。

根据上面 4 个步骤，可以建立如下程序：

```
function [t,y] = lineshootd(f1,f2,tspan,x0f,varargin); % 线性打靶法求解程序边值条件
为导数型
[t,y1] = ode45(f2,tspan,[1,0],varargin); % 计算函数y1(t)
[t,y2] = ode45(f2,tspan,[0,1],varargin); % 计算函数y2(t)
[t,y3] = ode45(f1,tspan,[0,0],varargin); % 计算函数y3(t)
M = [x0f(2)-x0f(1)*y2(end,2)-y3(end,2)]/y1(end,2); % 求参数M
[t,y] = ode45(f1,tspan,[m,x0f(1)],varargin); %求出原微分方程的解
```

参数说明：这个函数和前面定义的 lineshoot 基本相似，除参数 M 的定义不同外，其他参数相似，这里不再赘述。

例 12-10：求下面边值问题在区间[0,4]上的解，初值条件为：$y'(0)=0$，$y'(4)=0$。

$y''=2y'\cos t-y\sin 4t-\cos 3t$

分析：在这个问题中，边界条件使用导数来限定，因此需要调用函数 lineshootd 来计算这个微分方程。相应的程序也和例 12-9 类似，可以建立起相应的程序，计算程序如下：

```
f1 =inline('[y(2);2*cos(t)*y(2)-sin(4*t)*y(1)-cos(3*t)]','t','y');    % 定义原微分方程
f2 =inline('[y(2);2*cos(t)*y(2)-sin(4*t)*y(1)]','t','y');             % 定义其次微分方程
tspan = [0,4];                                                % 求解区间
x0f=[0,0];                                                   % 边值条件
[t,y] = lineshootd(f1,f2,tspan,x0f);                            % 调用函数 lineshootd 来求解
plot(t,y(:,1),t,y(:,2),'k:');
set(gca,'Fontsize',12);
set(gcf,'color','w');
```

上述程序所得结果如图 12.2 所示。

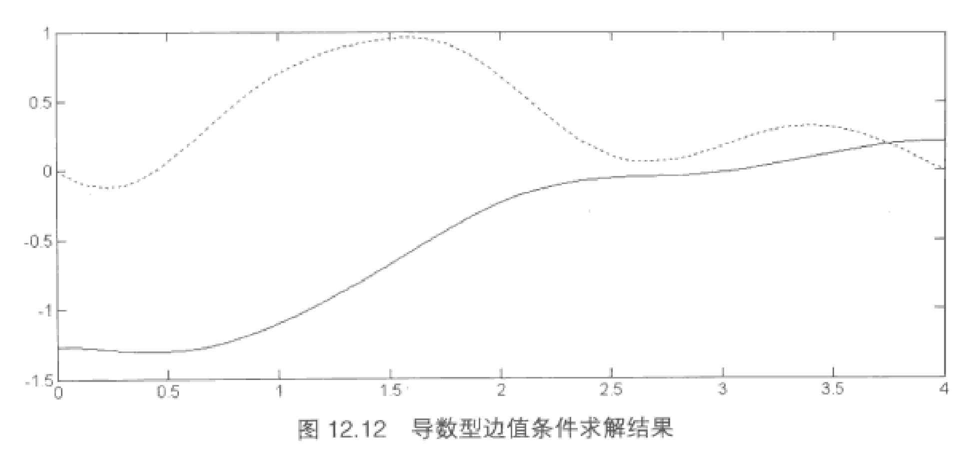


图 12.12　导数型边值条件求解结果

在图 12.12 中，实曲线对应着函数 $y(t)$，虚曲线表示 $y'(t)$。边界条件中 $y'(0)=0$，$y'(4)=0$，图 12.12 中两端点的函数 $y'(t)$ 都等于 0，从而说明这种方法求解的精度是很高的，也验证了程序的正确性。

前面介绍了两种类型的边界条件，即函数值和导数值的情况。此外还存在一类混合边界条件，即：$y(a)+c_1y'(a)=A$，$y(b)+c_1y'(b)=B$。

对于这类边界条件用户可以根据前面介绍的两种情况组合起来计算。感兴趣的读者可以考虑这个问题。

前面介绍了线性打靶法的求解过程，然而在实际问题中还存在着一些非线性微分方程，此时微分方程应该写为一般的形式：$x''=f(t,x,x')$，边值条件为 $x(a)=A$，$x(b)=B$。

现在需要求出一个参数 $m=x'(a)$，使得 $x(b)=B$ 成立，简记为 $x(b;m)=B$。这是一个复杂的超越方程，可以考虑引入牛顿迭代法求解参数 m。迭代关系式为：

$$m_{k+1}=m_k-\frac{x(b;m)-B}{\dfrac{\partial x(b;m)}{\partial m}}=m_k-\frac{v_1(b)-B}{v_3(b)} \quad (12\text{-}6)$$

其中$v_1 = x(t;m_k)$，$v_2 = x'(t;m_k)$，$v_3 = \partial x(t;m_k)/\partial m$，$v_4 = \partial x'(t;m_k)/\partial m$。通过上述关系可以建立关于$v_1$，$v_2$，$v_3$和$v_4$的方程。

$$\begin{cases} v_1' = v_2 \\ v_2' = f(t, v_1, v_2) \\ v_3' = v_4 \\ v_4' = v_3 \dfrac{\partial f(t, v_1, v_2)}{\partial x} + v_4 \dfrac{\partial f(t, v_1, v_2)}{\partial x'} \end{cases} \tag{12-7}$$

初始条件为：$v_1(a) = A$，$v_2(a) = m$，$v_3(a) = 0$和$v_4(a) = 1$。

在进行迭代求解时，用户可以任意给定m一个值，然后来求解微分方程组（12-7）在区间$[a,b]$上的解，把所得结果带入公式（12-6）中，直至$m_k - m_{k+1} = [v_1(b) - B]/v_3(b)$满足要求的精度为止。然后采用上面得到的$m$即可用求解初值方程来计算边值问题。

根据上面的分析可以建立相关的非线性打靶法程序，程序内容如下：

```
function [t,y] = nlshoot(f1,fv,tspan,xb,tol,varargin);       % 非线性打靶法求解微分方程
m=0;                   % m 的初值
m0=1;                  % 过程变量 m0
while abs(m-m0)>tol;
    m0=m;             % 更新过程量的数值
    [t,v] = ode45(fv,tspan,[xb(1),m,0,1],varargin);     % 计算迭代式
    m = m0-[v(end,1)-xb(2)]/v(end,3);              % 更新 m 的数值
end
[t,y] = ode45(f1,tspan,[xb(1),m],varargin);                % 利用得到的初值求解方程
```

用户可以利用上面的函数文件 nlshoot 求解一些带有边值条件的微分方程，其中参数 tol 用于控制参数 m 的误差，其他参数同前面的 lineshootd.m 文件。

例 12–11：求解下面的非线性边值问题。

$x'' = \cos x' \sin x$，边值条件为$x(0) = 1$，$x(6) = 2$。

分析：这里函数$f(t,x,x') = \cos x' \sin x$，对应的偏导数$\partial f(t,v_1,v_2)/\partial x = \cos x' \cos x$，$\partial f(t,v_1,v_2)/\partial x' = -\sin x' \sin x$。

这样可以得出方程组（12-7）中第 4 个方程为$v_4' = v_3 \cos v_2 \cos v_1 - v_4 \sin v_2 \sin v_1$，如此就可以很容易地建立函数 nlshoot 中的两个函数模型 f1 和 fv 了。求解程序如下：

```
f1 = inline('[x(2);cos(x(2))*sin(x(1))]','t','x');          % 定义函数 f1
fv =
inline('[v(2);cos(v(2))*sin(v(1));v(4);v(3)*cos(v(2))*cos(v(1))-v(4)*sin(v(2))
*sin(v(1))]','t','v');
[t,x] = nlshoot(f1,fv,[0,6],[1,2],1e-6);                 % 调用函数 nlshoot 求解非线性边值问题
plot(t,x(:,1),t,x(:,2),'k:');                          % 画图
set(gca,'Fontsize',12);
set(gcf,'color','w');
```

上述程序得到如图 12.13 所示的曲线。

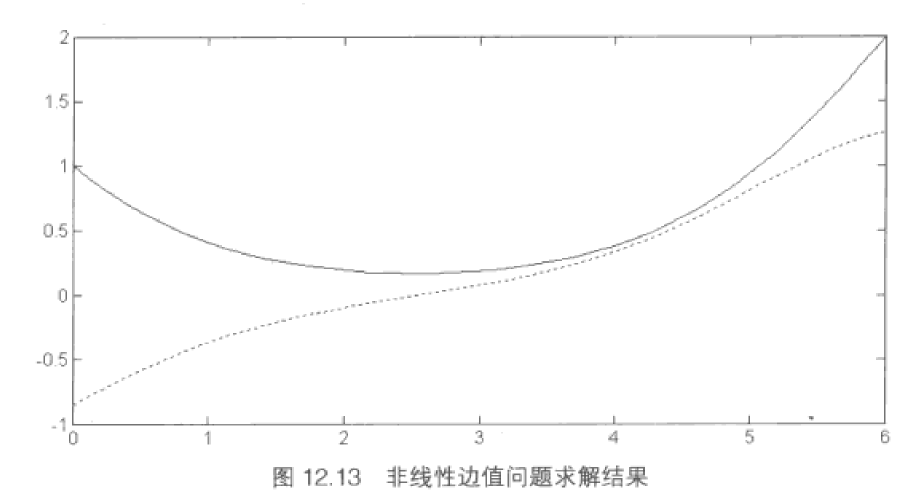


图 12.13　非线性边值问题求解结果

从图 12.13 中可以看出结果在边界上满足题目要求的条件。

12.6　时滞微分方程

时滞微分方程是指在微分方程表达式中含有一些滞后作用的项，即形如：

$$x' = f\left(t, x, x(t-t_1), x(t-t_2), ..., x(t-t_n)\right) \tag{12-8}$$

其中 $t_1, t_2, ..., t_n > 0$，是时间延迟常数。

在 MATLAB 中提供了函数 dde23 来求解这类方程，其调用格式为：

```
sol = dde23(ddefun,lags,history,tspan);
sol = dde23(ddefun,lags,history,tspan,options);
```

参数说明： sol 是输出的求解结果，其为结构体数据，sol.x 表示时间变量采样值，sol.y 表示函数值分量的取值。ddefun 表示时滞微分方程。lags 表示时间延迟常数。history 表示在求解时间区间上的变量值的函数，如果是一个函数形式，可以用 MATLAB 的函数文件形式定义，如果是变量则直接赋值即可。tspan 是求解时间范围。options 是控制过程参数的结构体，其由函数 ddeset 来设置，即 options=ddeset，用法类似函数 odeset。下面结合实例来介绍这个函数的使用。

例 12-12： 求解下面时滞微分方程的解。

$$\begin{cases} x_1'(t) = 0.5x_3(t-3) + 0.5x_2(t)\cos t \\ x_2'(t) = 0.3x_1(t-1) + 0.7x_3(t)\sin t \\ x_3'(t) = x_2(t) + \cos 2t \end{cases}$$

当 $t \le 0$ 时，$x_1(t) = 0$，$x_2(t) = 0$，$x_3(t) = 1$。

分析： 在这个方程中，函数 $x_1(t)$ 和 $x_3(t)$ 存在着不同的时间延迟，因此需要考虑使用函数 dde23 来求解。方程自身就是一阶微分方程组的形式，不必再进行转化，根据关系建立起各个参数即可进行求解。定义时滞微分方程的函数文件内容如下：

```
function dx = ddefun1(t,x,lags);
x1d = lags(:,1);        % y 提取第一延时常数对应的列向量
x3d = lags(:,2);        % y 提取第二延时常数对应的列向量
dx = [0.5*x3d(3)+0.5*x(2)*cos(t);...
```

```
    0.3*x1d(1)+0.7*x(3)*sin(t);...
    x(2)+cos(2*t)];      % 定义微分方程
```

用户可以根据上面的例子对照原微分方程组来理解参数的对应关系。调用下面的程序即可求解这个时滞微分方程。

```
lags = [1,3];                    % 定义延时常数
history = [0,0,1];                 % 历史数据
tspan = [0,8];                   % 求解时间范围
sol = dde23('ddefun1',lags,history,tspan);  % 调用 dde23 函数求解
t = sol.x;                      % 提取变量采样数据
x = sol.y;                      % 提取函数分量的离散值
subplot(131);plot(t,x(1,:));        % 绘制 x1(t)对应的曲线
xlabel('\itt');ylabel('{\itx}_1');      % 标注
subplot(132);plot(t,x(2,:));        % 绘制 x2(t)对应的曲线
xlabel('\itt');ylabel('{\itx}_2');      % 标注
subplot(133);plot(t,x(3,:));        % 绘制 x3(t)对应的曲线
xlabel('\itt');ylabel('{\itx}_3');      % 标注
set(gcf,'color','w','Position',[251 378 959 252]);  % 重置图形窗口的背景色、位置和大小
```

所得图形如图 12.14 所示。这里在 3 个坐标轴内给出了 $x_1(t)$，$x_2(t)$和 $x_3(t)$相应的曲线，其中 $x_1(t)$初始一段时间内发生振荡现象，随后逐渐增大，而 $x_2(t)$和 $x_3(t)$初始阶段数值增加，随后函数值很快衰减。

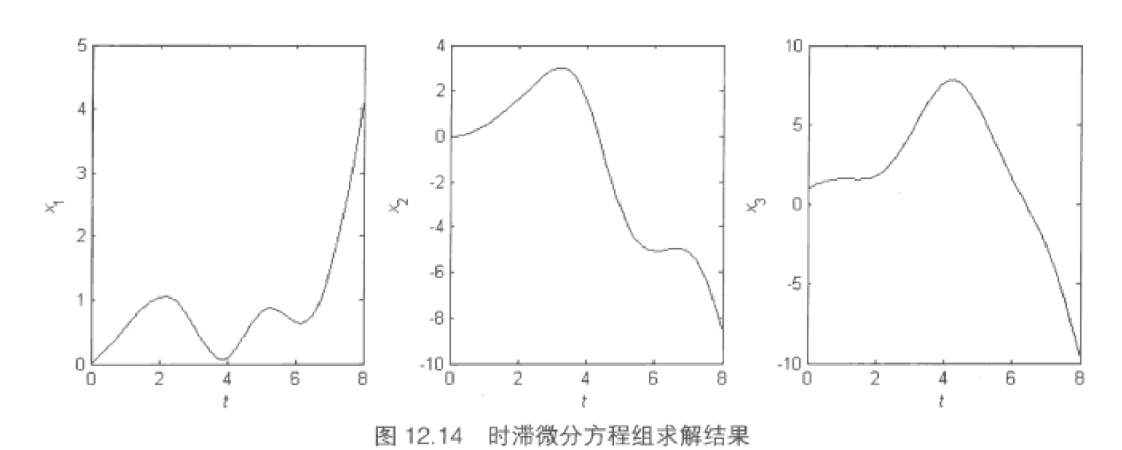


图 12.14　时滞微分方程组求解结果

12.7　偏微分方程

在描述流体运动和场的分布的大多数数学模型时，微分方程模型是根据质量连续和能量守恒等基本原理推导出来的，再加上适当的初始条件或者边界条件所构成。在一维问题中，它们是常微分方程；而在二维或三维问题中，它们就是偏微分方程。本节来介绍偏微分方程的求解方法。

MATLAB 提供了 PDE 工具箱来专门求解一些典型的偏微分方程，如椭圆形方程、抛物形方程、双曲线方程、特征值方程、椭圆形方程组以及非线性椭圆形方程等。首先来介绍一下 PDE 工具箱的工作界面。

用户在命令窗输入 pdetool 即可调出对话窗口，如图 12.15 所示。下面介绍菜单中各部分功能：

File 菜单功能介绍如表 12.3 所示，Options 菜单功能介绍如表 12.4 所示，Draw 菜单功能介绍如表 12.5 所示。

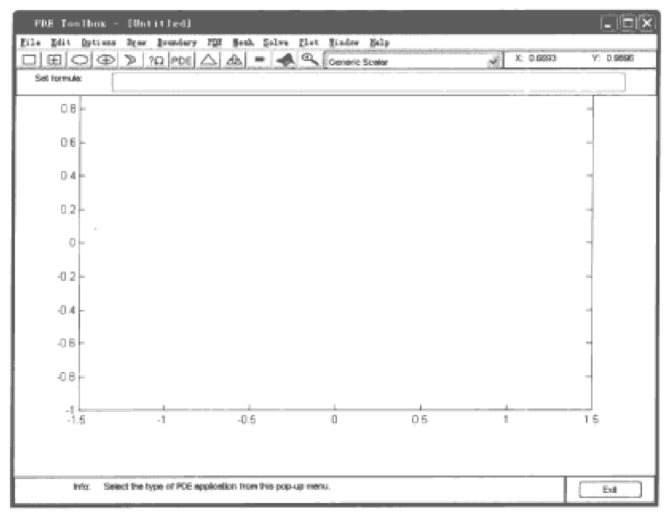


图 12.15　PDE 工具箱窗口

表 12.3　File 菜单功能介绍

菜单名	功能	菜单名	功能
New	新建模型	Save As	保存结果到另一个文件夹中
Open	加载 M 文件	Print	打印图形
Save	保存结果	Exit	退出界面

在 Edit 菜单中含有一些基本的 Windows 操作，如撤销（Undo）、剪切（Cut）、复制（Copy）、粘贴（Paste）、删除（Clear）、全选（Select All）等。

表 12.4　Options 菜单功能介绍

菜单名	功能	菜单名	功能
Grid	开启/关闭栅格	Turn off Toolbar	关闭工具栏按钮的帮助文档
Grid Spacing	调整栅格大小	Zoom	图形缩放的开启和关闭
Snap	捕捉栅格开启/关闭	Application	应用模式
Axis Limits	改变坐标轴比例	Refresh	重新显示图中实体
Axis Equal	控制 X 轴和 Y 轴的比例关系		

表 12.5　Draw 菜单功能介绍

菜单名	功能
Draw Mode	开始绘图模式
Rectangle/Square	角点方式画矩形（操作：Ctrl+鼠标）
Rectangle/Square （centered）	中心方式画矩形（操作：Ctrl+鼠标）
Ellipse/Circle	矩形角点方式画椭圆（操作：Ctrl+鼠标）
Ellipse/Circle （centered）	中心方式画椭圆（操作：Ctrl+鼠标）
Polygon	画多边形，右键自动封闭多边形
Rotate	旋转选定实体
Export Geometry Description, Set Formula, Labels	导出几何描述、公式、标注等信息到 workspace

Boundary 菜单功能介绍如表 12.6 所示，PDE 菜单功能介绍如表 12.7 所示，Mesh 菜单功能介绍如表 12.8 所示，Solve 菜单功能介绍如表 12.9 所示，Plot 菜单功能如表 12.10 所示。

表 12.6　Boundary 菜单功能介绍

菜单名	功能
Boundary Mode	开始边界模式
Specify Boundary Conditions	指定为已选边界按已输入的边界条件处理
Show Edge Labels	显示边界区域标注
Show Subdomains Labels	显示子区域标注
Remove Border Subdomain	删除已选取的子区域边界
Export Decomposed Geometry, Boundary Cond's	导出几何矩阵和边界条件矩阵到 workspace
Specify boundary conditions	指定边界条件

表 12.7　PDE 菜单功能介绍

菜单名	功能
PDE Mode	开始偏微分方程模式
Show Subdomain Labels	显示子区域标注
PDE Specification	输入 PDE 参数和类型
Export PDE Coefficents	当前 PDE 参数输出到 workspace

表 12.8　Mesh 菜单功能介绍

菜单名	功能	菜单名	功能
Mesh Mode	开始网格模式	Display Triangle Quality	演示三角形网格的质量
Initialize Mesh	初始化三角形网格	Show Node Labels	显示网格节点标注
Refine Mesh	加密当前三角形网格	Show Triangle Labels	显示三角形网格标注
Jiggle Mesh	优化网格	Expoert Mesh	导出网格参数到 workspace
Undo Mesh Change	返回到前一次网格模式	Parameters	参数设置

表 12.9　Solve 菜单功能介绍

菜单名	功能
Solve PDE	求解当前设定的偏微分方程
Parameters	参数设置
Export Solution	导出求解结果到 workspace

表 12.10　Plot 菜单功能介绍

菜单名	功能
Plot Solution	画出求解结果
Parameters	设置绘图参数
Export Movie	若动画被录制，则动画矩阵将输出到 workspace

从 Window 菜单中，用户可以选择当前打开的所有 MATLAB 图形窗口，或选择后面的窗口到最前面。在 Help 菜单中，可以选择简洁帮助窗口和带有一些程序信息的窗口。

在主菜单下面，一些图标按钮（如图 12.16 所示）可以快捷地运行 PDE 函数和各菜单中的主要功能。按钮的功能从左到右按序排列：左边 5 个按钮为绘图模式，其余 6 个为边界、网格、解方程和图形显示控制功能，最右边的是图形缩放功能键。具体功能如表 12.11 描述。



图 12.16　PDE 工具箱中的按钮

表 12.11　按钮功能介绍

序号	功能	序号	功能
1	角点方式画矩形（操作：Ctrl+鼠标）	7	打开 PDE 对话框
2	中心方式画矩形、方形（操作：Ctrl+鼠标）	8	初始化三角形网格
3	矩形角点方式画椭圆（操作：Ctrl+鼠标）	9	加密三角形网格
4	中心方式画椭圆/圆（操作：Ctrl+鼠标）	10	解偏微分方程
5	画多边形，右键自动封闭多边形	11	画出解的三维图形
6	开始界面模式	12	缩放图形

下面介绍典型微分方程和边界条件。

- 椭圆形方程：$-\nabla\cdot(cu)+au=f$
- 抛物形方程：$d\,\partial u/\partial t-\nabla\cdot(c\nabla u)+au=f$
- 双曲线方程：$d\,\partial^2 u/\partial t^2-\nabla\cdot(cu)+au=f$
- 特征值方程：$-\nabla\cdot(c\nabla u)+au=\lambda du$

Ω 是平面有界区域，c，a，f，d 以及未知函数 u 是定义在 Ω 上的实或复函数。下面以椭圆形偏微分方程为例说明 PDE 工具箱的使用。

例 12-13： 求下面的椭圆形偏微分方程在矩形区域（$0\le x\le 3$，$0\le y\le 4$）内的解。

$\dfrac{\partial^2 U}{\partial x^2}+\dfrac{\partial^2 U}{\partial y^2}=0$，边界条件为：$U\big|_{x=0}=0$，$U\big|_{y=0}=x$，$U\big|_{x=3}=\sin\dfrac{\pi y}{2}$，$\dfrac{\partial U}{\partial n}\bigg|_{y=4}=0$。

分析： 这是一个椭圆形偏微分方程，用户可以调用 PDE 界面上的组件来求解这个微分方程。下面详细介绍这个方程的求解步骤。求解过程如下：

step 1 设定方程类型。单击图 12.16 中第 7 个按钮选择方程类型并输入参数，如图 12.17 所示，最后单击 OK 按钮确认。

step 2 设置区域。单击图 12.16 中的第 1 个按钮来设置求解区域，得出一个矩形之后在矩形内部双击左键，在相应的位置填好参数，如图 12.18 所示。

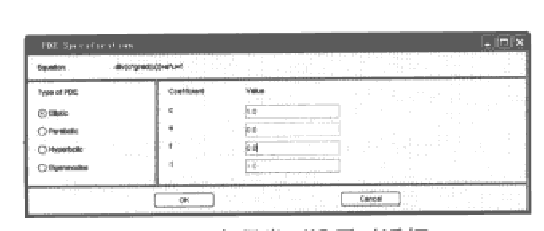


图 12.17　方程类型设置对话框

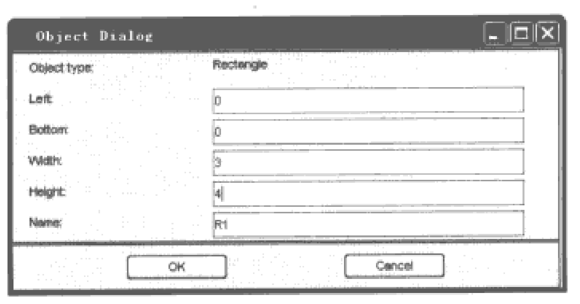


图 12.18　实体的参数设置对话框

其中，Left 指矩形区域最左侧的坐标值，Bottom 指最下面的坐标值，Width 指矩形区域的宽度，Height 指矩形区域的高度，Name 是矩形对象的标注（这一项用户可以使用默认值）。设置好参数之后单击 OK 按钮可以确认区域的定义。

step 3 输入边界条件。用户可以单击图 12.16 中的第 6 个按钮，此时矩形边界变为红色边界线，用户分别用左键双击 4 条线来设置边界条件。其中 $U|_{x=0}=0$，$U|_{y=0}=x$，$U|_{x=3}=\sin\frac{\pi y}{2}$ 是狄利克莱（Dirichlet）条件，参数 h 等于 1，r 分别对应于 0，x，sin(pi*y/2)，而 $\left.\frac{\partial U}{\partial n}\right|_{y=4}=0$ 是诺依曼（Neumann）条件，参数 g=0，q=0。输入的界面如图 12.19 所示。用户按条件正确输入就可以了。

图 12.19　边界条件输入对话框

step 4 单击图 12.16 中第 8 个按钮，可以得到三角形剖分网格。其由蓝色线段构成，这里略去，感兴趣的用户可以深入研究。

step 5 用户单击图 12.16 中的第 10 按钮可以进行偏微分方程的求解，同时 MATLAB 会利用默认设置把结果显示出来。

step 6 用户可以单击图 12.16 中的第 11 按钮来绘制结果的曲面图。单击这个按钮之后会出现一个对话框，如图 12.20 所示，在这个对话框中可以进行参数的设置。设置完毕后单击 Plot 按钮就可以画出图形来了，如图 12.21 所示。

图 12.20　绘图参数对话框

这里补充一下关于坐标轴范围设置的技巧，用户可以选择菜单 Options/Axes Limits 弹出如图 12.22 所示的对话框。

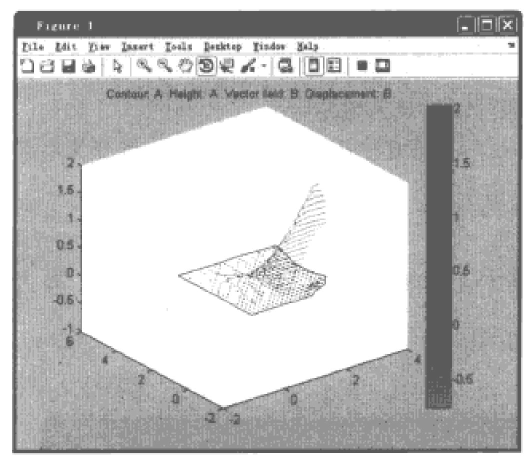


图 12.21 绘图结果

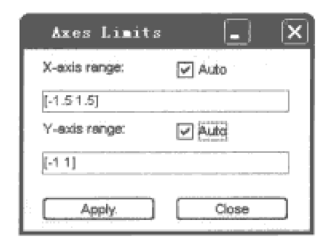


图 12.22 坐标轴范围设置对话框

这里用户可以勾选两个 Auto 复选框，MATLAB 将会自动地根据坐标轴内的对象范围来选择坐标轴范围。

除了前面介绍的 PDE 工具箱的界面形式，MATLAB 还提供了函数 pdepe 来求解一类形如下式的偏微分方程：

$$c(x,t,u,\partial u/\partial x)\frac{\partial u}{\partial t}=x^{-m}\frac{\partial}{\partial x}\left[x^{m}f(x,t,u,\partial u/\partial x)\right]+s(x,t,u,\partial u/\partial x) \tag{12-9}$$

这样的微分方程的定义需要按下面的格式：

```
function [c,f,s] = pdefunname(x,t,u,ux)
```

用户需要建立一个名为 pdefunname.m 的函数文件，其中 ux 对应于$\partial u/\partial x$，这个函数文件将作为 pdepe 的输入函数中的偏微分方程的定义。边界条件的一般描述是：

$$p(x,t,u)+q(x,t,u)f(x,t,u,\partial u/\partial x)=0 \tag{12-10}$$

公式（12-10）需要建立函数文件 pdebcname.m 来定义，其输入和输出参数如下面的形式：

```
function [pa,qa,pb,qb] = pdebcname(x,t,u,ux)
```

其中输出参数 pa 和 qa 为 a 端点的值，输出参数 pb 和 qb 为 b 端点的值。

而初始条件可以描述为$u(x,t_0)=u_0$，用户编写下面形式的函数文件即可：

```
function u0 = pdeicname(x);
```

下面给出函数 pdepe 帮助文档中例子的改编版本来说明该函数的使用，程序代码如下：

```
clc;
clear;
close all;
x = linspace(0,1,30);           % 定义变量 x 的采样点
t = linspace(0,2,30);           % 定义变量 t 的采样点
sol = pdepe(0,@pdex1pde,@pdex1ic,@pdex1bc,x,t);   % 调用函数 ddepe 求解偏微分方程
mesh(x,t,sol);                 % 画图
colormap([0,0,0]);             % 设置网格颜色为黑色
set(gcf,'Color','w');           % 设置背景色为白色
```

函数 pdex1pde，pdex1ic 和 pdex1bc 为 MATLAB 自带函数，其表达式用户可参阅帮助文档，

所得图形如图 12.23 所示。该图形关于位置 x=0.5 平面对称，同时随着参数 t 的增加函数值的起伏程度降低。

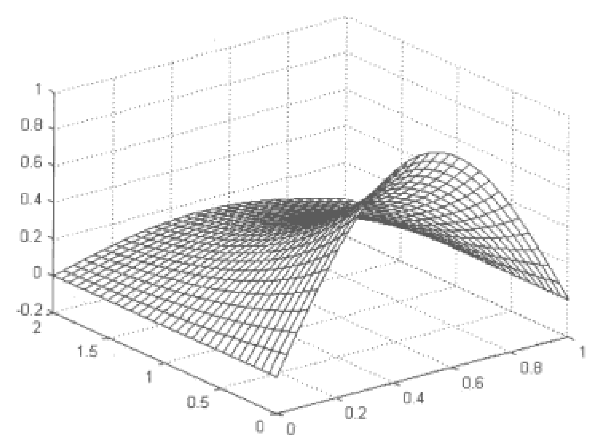


图 12.23 利用函数 pdepe 求得偏微分方程结果

12.8 利用微分算积分

一般地，积分表达式可以表示为：

$$I=\int_a^b f(x)\mathrm{d}x \tag{12-11}$$

上面的积分也可以表示为 $I=F(b)-F(a)$，其中 $F(x)$ 和 $f(x)$ 之间的关系是：

$$\frac{\mathrm{d}F(x)}{\mathrm{d}x}=f(x) \tag{12-12}$$

因此只要求出函数 $F(x)$，用户就可以根据 $I=F(b)-F(a)$ 来计算这个积分结果。而求解式（12-12）可以使用 ode 系列函数来计算 $F(x)$ 在区间 $[a,b]$ 上的数值解，求出所得结果在两端点的差值就可以得到积分结果，其中初值用户可以自行设置而不影响求解结果。下面举例说明计算方法。

例 12-14： 计算下面的定积分结果。

$$i_1=\int_0^1\frac{1+x^2}{1+x^3}\sin x\mathrm{d}x\text{ ，}\quad i_2=x\sin x+\int_0^2 i_2(x)\mathrm{d}x\text{ ，}\quad i_3'=i_3x\sin x+\int_0^3 i_3(x)\mathrm{d}x$$

分析： 第一个积分可以直接通过求解微分方程 $I_1'=\frac{1+x^2}{1+x^3}\sin x$（求解区间为 $[0,1]$）得到。而第二个积分对应的微分方程可以通过两边对 x 求导数来得到，即 $I_2'=\sin x+x\cos x$，求解区间为 $[0,2]$，其中初始条件随意设定。

根据上面的分析可以建立如下程序来计算微分方程，为方便起见，这里使用函数 inline 来定义微分方程。程序代码如下：

```
I1 = inline('(1+x^2)/(1+x^3)*sin(x)','x','I1');    % 建立微分方程
I2 = inline('sin(x)+x*cos(x)','x','I2');           % 建立微分方程
[x,I1] = ode45(I1,[0,1],2);         % 调用函数 ode45 解微分方程
[x,I2] = ode45(I2,[0,2],2);         % 调用函数 ode45 解微分方程
```

```
I1 = I1(end)-I1(1)
I2 = I2(end)-I2(1)
```

输出结果为：

```
I1 =    0.4949
I2 =    1.8186
```

更换第一个微分方程和第二个微分方程的初始条件，所得结果并不改变，这证明了前面的观点。此外用户可以用 options = odeset('RelTol',v)来设置求解精度，其中 v 是一个较小的正数。

12.9 小结

微分方程模型是科学研究中经常遇到的问题，本章主要介绍了微分方程的求解方法。首先介绍了极限的符号求解方法，接下来介绍了利用函数 diff 和 jacobian 计算函数的全导数。函数 dsolve 可以用来计算一些简单的微分方程的解析解。12.4 节详细介绍了 MATLAB 中 ode 系列函数的用法，通过实例给出了不同类型微分方程的解法。本章在后面部分分别介绍了利用打靶法求解边值问题，其中包括两种类型边值条件的微分方程的解法；介绍了利用函数 dde23 来求解时滞微分方程；结合偏微分工具箱介绍了利用 MATLAB 求解偏微分方程，以及如何用函数 pdepe 来求解一类偏微分方程；最后介绍了利用求解微分的方法来计算数值积分。

◆ ◆ ◆

第 13 章 积分运算

本章包括

- ◆**级数求和** 介绍利用函数 symsum 级数求和的方法。
- ◆**离散积分的计算** 介绍计算积分的 MATLAB 自带的求解积分函数以及一些数值算法。
- ◆**奇异积分计算** 介绍一些含有奇异点的被积函数对应的积分计算。

微积分思想是高等数学的基石，很多科学问题中都存在着不同类型的积分公式模型。积分的计算在很多学科中都有着重要的应用。本章将介绍利用 MATLAB 求解积分的问题。在 MATLAB 的库函数中，提供了两种方式来计算积分问题，即符号积分和数值积分。对于一些简单的函数，用户可以进行符号积分运算来求解结果，而数值积分可以用来求解不同类型的积分。人们研究了多种数值积分方法，在实际应用中需要根据被积函数的性质来选择数值积分方法。本章重点介绍一些数值积分算法，帮助用户解决常见的积分问题。

13.1 级数求和

级数是一系列与自然数 n 有关的函数集合，其可以表示为：

$$a_n = f(n) \tag{13-1}$$

这里 $f(n)$ 是一个与 n 有关的函数。n 是不连续的，其步长为 1，而级数求和可以认为是一种比较粗糙的积分模型，即积分步长 dn 不是趋于 0，而是等于 1。

13.1.1 symsum 函数

在 MATLAB 中提供了 symsum 函数来计算级数求有限项或者无穷项的和，其调用格式为：

```
s = symsum(S);
s = symsum(S,v);
s = symsum(S,a,b);
s = symsum(S,v,a,b);
```

参数说明： s 是返回的计算结果。S 是级数的表达式。v 用来指定求和的符号变量。a 和 b 用来指定对[a,b]区间内的自然数求和，其中 a 和 b 可以是负数或者小数。当 a 或者 b 为小数时，MATLAB 将它们向零取整，即 fix(a)或者 fix(b)，这一点用户在使用时需要特别注意。

下面举例说明函数 symsum 的用法。

```
syms m n
s1 = symsum(1/n,1,inf)
s2 = symsum(1/n^2,1,inf)
s3 = symsum(1/n^m,1,inf)
```

输出为：

```
s1 =Inf
s2 =1/6*pi^2
s3 =sum(1/(n^m),n = 1 .. Inf)
```

可见在计算 $1/n^m$ 时，MATLAB 不能给出一个数值结果，但当 m 为已知数时可以得到一个确定的值。在数学上，有如下关系式：

$$\sum_{n=1}^{\infty}\frac{1}{n^k}=\text{zeta}(k)$$

在 MATLAB 中用户可以直接调用函数 zeta 来计算此类级数无穷项之和，调用格式为：

```
y = zeta(k)
```

此外还可以用函数 symsum 来计算含变化参数的级数，比如求级数 $a_n=\frac{(-m)^n}{n}$ 的无穷项之和，n 从 1 到无穷。程序如下：

```
s4 = symsum((-m)^n/n,1,inf)
```

输出结果为：

```
s4 =-log(1+m)
```

13.1.2 taylor 函数

这里再介绍关于泰勒级数展开的函数 taylor，其调用格式为：

```
t = taylor(f);
t = taylor(f,n);
t = taylor(f,n,x0);
t = taylor(f,v);
```

参数说明： t 是返回的泰勒级数。f 是函数的符号表达式。n 为整数时，MATLAB 将进行麦克劳林级数展开而得到 n-1 阶多项式，n 的默认整数值是 6。当 n 为一个小数时，返回在 n 处展开的一个次多项式形式。x0 表示在 x0 处展开多项式。v 用来指定对符号变量 v 进行泰勒级数展开。

例 13–1： 调用该函数求解代数函数 $f(x,t)=t\sin x$。

```
syms x t;
f = sin(x)*t; % 输入函数的表达式
t1 = taylor(f)       % 计算 0 点处的泰勒级数
t2 = taylor(f,6)     % 计算 0 点处的 6 次多项式展开泰勒级数
t3 = taylor(f,6,3)   % 计算在 x=3 处的 6 次多项式展开泰勒级数
t4 = taylor(f,t)     %计算 0 点处的泰勒级数，变量名为 t
t5 = taylor(f,4.1)   % 选择输入参数为一个小数的情况
```

输出为：

```
t1 =x*t-1/6*t*x^3+1/120*t*x^5
t2 =x*t-1/6*t*x^3+1/120*t*x^5
t3
=sin(3)*t+cos(3)*t*(x-3)-1/2*sin(3)*t*(x-3)^2-1/6*cos(3)*t*(x-3)^3+1/24*sin(3)
```

```
*t*(x-3)^4+1/120*
cos(3)*t*(x-3)^5
t4 =sin(x)*t
t5 =
sin(41/10)*t+cos(41/10)*t*(x-41/10)-1/2*sin(41/10)*t*(x-41/10)^2-1/6*cos(41/10
)*t*(x-41/10)^3+1/24*sin(41/10)*t*(x-41/10)^4+1/120*cos(41/10)*t*(x-41/10)^5
```

对于给定的周期为T的函数$f(x)$可以存在傅里叶级数。对于定义在区间$[a,a+T]$的非周期函数$g(x)$，用户可以进行周期解析延拓，从而得到一个周期函数。定义在区间$[0,T]$的函数$f(x)$的傅里叶级数可以表示式为：

$$f(x)=\frac{a_0}{2}+\sum_{n=1}^{\infty}a_n\cos\frac{2n\pi x}{T}+b_n\sin\frac{2n\pi x}{T} \quad (13\text{-}2)$$

其中系数a_n和b_n可以表示为：

$$a_n=\frac{2}{T}\int_0^{2T}f(x)\cos\frac{2n\pi x}{T}\mathrm{d}x\,,\quad n=0,1,2,\ldots \quad (13\text{-}3)$$

$$b_n=\frac{2}{T}\int_0^{2T}f(x)\sin\frac{2n\pi x}{T}\mathrm{d}x\,,\quad n=1,2,\ldots \quad (13\text{-}4)$$

可以看出只要系数a_n和b_n确定了就可以得到傅里叶级数的结果了。

13.1.3 傅里叶级数

在MATLAB自带函数中，没有函数来计算傅里叶级数的系数，用户需要自己编写程序来计算。下面的函数文件Fseries.m可以用来计算。

```
function [an,bn] = Fseries(f,T,n);
an(1) = quadl(inline(f),0,T);                        % 计算a0
for k=1:n;
    fc = inline(['(',f,')','.*cos(',num2str(2*k*pi/T),'*x)']);  % 定义被积函数
    fs = inline(['(',f,')','.*sin(',num2str(2*k*pi/T),'*x)']);  % 定义被积函数
    an(k+1) = quadl(fc,0,T);          % 调用积分函数quadl计算系数an
    bn(k) = quadl(fs,0,T);            % 调用积分函数quadl计算系数bn
end
```

参数说明： an和bn是返回的傅里叶系数数值，其中$a_n=[a_0,a_1,\ldots,a_n]$，$b_n=[b_1,\ldots,b_n]$。f是原函数在区间$[0,T]$内表达式对应的字符串，需要说明的是f中需要支持对变量x进行向量计算。T是函数的周期。n表示展开的项数，总项数为2n+1。在这个函数的定义中用到MATLAB函数quadl，该函数的详细用法将在后面的章节详细介绍。

下面的程序是计算函数$f(x)=x\tanh x$在区间$[0,3]$的傅里叶级数：

```
f = 'x.*tanh(x)';
[an,bn] = Fseries(f,3,5)
```

输出为：

```
an =    2.3093   -0.2510   -0.0656   -0.0268   -0.0146   -0.0092
bn =   -0.3590   -0.2334   -0.1583   -0.1188   -0.0950
```

13.2 离散积分的计算

本节来介绍积分问题的求解方法。首先来介绍 MATLAB 提供的一些计算积分的函数，如 int，quad，quadl 等，然后介绍一些典型的数值积分算法。

13.2.1 函数法

在 MATLAB 中，提供了两种计算积分的函数，一类是符号积分，由函数 int 来实现；另一类是数值积分，由 quad 和 quadl 等函数来实现。下面详细介绍这些函数的使用。

13.2.1.1 函数 int

函数 int 可以得到积分问题的符号解，包括不定积分和定积分两种形式的积分，其调用格式为：

```
I = int(S)                % 格式 1
I = int(S,v)              % 格式 2
I = int(S,a,b)            % 格式 3
I = int(S,v,a,b)          % 格式 4
```

参数说明：I 是输出的积分结果，其为一个符号型数据，如果用户希望得到一个 double 型数据，可以使用函数 double 转化。S 为被积函数的符号表达式或者用符号表达式构成的矩阵。v 用于指定积分变量。a 和 b 用来定义积分的上下限。

其中格式 1 和格式 2 用来计算不定积分，而格式 3 和格式 4 用来计算定积分。下面举例介绍 int 函数的用法。

例 13-2：求下面的函数在指定区间的积分结果：$f_1(x)=\dfrac{x^3}{1+x^4}$，计算不定积分；$f_2(x)=\sin ax$，计算不定积分；$f_3(x)=\sin(2\cos x)$，积分区间为 $[0,\pi/2]$；$f_4(x)=\cos(a\sin x)$，积分区间为 $[0,\pi/2]$；$f_5=\begin{bmatrix} x & x\cos ax \\ \dfrac{1}{2+\sin x} & \dfrac{3+\sin x}{2+\cos x}\end{bmatrix}$，积分区间为 $[0,\pi]$。

分析：本问题中对 $f_1(x)$ 和 $f_2(x)$ 计算的是不定积分，可以使用函数 int 的第 1 种调用格式和第 2 种调用格式。而对 $f_3(x)$ 和 $f_4(x)$ 计算的是定积分，此时需要调用的是第 3 种调用格式和第 4 种调用格式。函数 $f_5(x)$ 是一个 2×2 的符号函数矩阵，需要计算其定积分。相应的求解积分程序如下：

```
syms x a;
f1 = x^3/[1+x^4];    % 定义函数 f1(x)
f2 = sin(a*x);       % 定义函数 f2(x)
f3 = sin(2*cos(x));   % 定义函数 f3(x)
f4 = cos(a*sin(x));   % 定义函数 f4(x)
f5 = [x,x*cos(a*x);1/[2+sin(x)],[3+sin(x)]/[2+cos(x)]]; % 定义符号函数 f 矩阵 5(x)
F1 = int(f1)
F2 = int(f2)
F3 = int(f3,0,pi/2)
F4 = int(f4,x,0,pi/2)
F5 = int(f5,x,0,pi)
```

输出积分结果为：

```
F1 =1/4*log(1+x^4)
F2 =-1/a*cos(a*x)
F3 =1/2*pi*StruveH(0,2)
F4 =1/2*pi*besselj(0,a)
F5 =[   1/2*pi^2, (-1+cos(pi*a)+sin(pi*a)*a*pi)/a^2]
[  2/9*pi*3^(1/2),            log(3)+pi*3^(1/2)]
```

可见 MATLAB 可以快捷地给出一些函数的不定积分和定积分结果，利用 MATLAB 的这个功能可以节省一些公式的理论推导工作。

众所周之，不定积分的结果要出现一个常数，而 MATLAB 得到的结果中这个函数缺省了；在被积函数中存在多个符号变量时，如果用户不指定积分变量，则 x 被认为是积分变量；一些积分问题的结果可能会与某些特殊函数有关系，MATLAB 给出一个利用特殊函数表达的结果，前面积分结果中的 StruveH 和 besselj 是两个特殊函数，用户在命令窗中输入"mhelp StruveH"和"help besselj"可以查看它们的帮助信息，使用函数 double 作用于积分结果，即 double(F3)和 double(F4)就可以得到一个数值了；当被积函数不存在解析解时，MATLAB 将输出一个积分表达式，比如计算函数 $\cos(a\sin x)$ 在区间 $[0,1]$ 的定积分，即:

```
syms x a
int(cos(a*sin(x)),0,1)
```

执行后在命令窗会提示用户：

```
Warning: Explicit integral could not be found.
> In sym.int at 58
ans =int(cos(a*sin(x)),x = 0 .. 1)
```

此时用户需要考虑使用数值方法来计算积分问题，积分的上下限还可以使用 inf 和-inf 来表示正负无穷大，积分限还可以用符号变量来表示，如此可以计算变上（下）限积分以及一些复杂的上下限形式。

例 13–3：求下面的积分。

$$g_1(x)=\int_0^{\sin t}\frac{\sin x}{x}\mathrm{d}x$$

$$g_2(x)=\int_{\cos t}^{\sin t}\sqrt{1+x^2}\,\mathrm{d}x$$

分析：这两个积分是变上限积分，其中 $g_2(x)$ 的上下限是三角函数。对于这样的积分，用户仍可以利用函数 int 来求解。在对应的上下积分限把三角函数形式写进去就可以了。

相应的计算程序如下：

```
syms x t;
g1 = int(sin(x)/x,x,0,sin(t))
g2 = int(sqrt(1+x^2),x,cos(t),sin(t))
```

输出结果为：

```
g1 = sinint(sin(t))
```

```
g2
=-1/2*cos(t)*(1+cos(t)^2)^(1/2)-1/2*asinh(cos(t))+1/2*sin(t)*(1+sin(t)^2)^(1/2
)+1/2*asinh(sin(t))
```

其中 asinh 是反双曲正弦函数，而函数 sinint 是特殊函数，其定义为 sinint(x)=int (sin(t)/t,t,0,x)。可见 MATLAB 对于第一个变上限积分没有计算，而是以一个特殊函数给出来。

例 13-4： 利用函数 int 来计算下面函数对所有变量的多重定积分。

$h_1(x,y)=\sin(xy^2)$，变量取值范围是 $x,y\in[0,1]$。

$h_2(x,y,z)=\sin(x^2y)+x\sin z$，变量取值范围是 $x,y\in[0,1]$， $z\in[0,2]$。

$h_3(x,y,z)=xy\cos z$，变量取值范围是 $x\in[0,1]$， $x+y\in[0,1]$， $z\in[0,1]$。

分析： 上述 3 个函数的多重积分，用户可以调用函数逐个变量求积分。前两个函数用户在求积分的时候可以以任意顺序对多个变量积分，在第 3 个函数中，变量 x 和 y 之间不是独立的，存在着一定的关系，用户需要按照顺序来计算，而变量 z 是独立的。在计算函数 $h_3(x)$ 的积分中，可以按 $y\to x\to z$ 的顺序来求解积分，其中变量 y 的积分范围是 $[0,1-x]$，而变量 x 和 z 的积分范围是 $[0,1]$。求解程序如下：

```
syms x y z;
h1 = int(sin(x*y^2),x,0,1);         % 对变量 x 积分
h1 = int(h1,y,0,1)                  % 对变量 y 积分
h2 = int(sin(x^2*y)+x*sin(z),x,0,1);    % 对变量 x 积分
h2 = int(h2,y,0,1);                 % 对变量 y 积分
h2 = int(h2,z,0,2)                  % 对变量 z 积分
h3 = int(x*y*cos(z),y,0,1-x);       % 对变量 y 积分
h3 = int(h3,x,0,1);                 % 对变量 x 积分
h3 = int(h3,z,0,1)                  % 对变量 z 积分
```

上述程序输出为：

```
h1 =-1+cos(1)+2^(1/2)*pi^(1/2)*FresnelS(2^(1/2)/pi^(1/2))
h2 =1/2+1/3*hypergeom([3/4, 1],[3/2, 7/4, 2],-1/4)-1/2*cos(2)
h3 =1/24*sin(1)
```

其中 FresnelS 和 hypergeom 是特殊函数，用户在命令窗输入 mhelp FresnelS 和 mhelp hypergeom 可以查阅这两个函数的帮助信息。使用下面的语句可以得到相应的数值解：

```
V123 = double([h1,h2,h3])
```

输出为：

```
V123 =    0.1608    1.0298    0.0351
```

在第 12 章介绍了利用 MATLAB 求解微分方程数值解的函数来计算积分问题，本节考虑使用 MATLAB 中的函数 dsolve 来计算 MATLAB 的积分问题，其原理也是求被积函数对应的微分方程的解，即下面的微分方程：

$$\frac{\mathrm{d}F(x)}{\mathrm{d}x}=f(x) \tag{13-5}$$

其中 $f(x)$ 表示被积函数的表达式。$F(x)$ 是对应的原函数。对于不定积分，函数 dsolve 得到的结果就是积分的结果，而定积分的计算可以通过积分限的差值来得到积分结果。函数 dsolve 的

用法已经在第 12.3 节进行了介绍。下面举例说明利用函数 dsolve 计算积分问题。

例 13–5： 利用函数 dsolve 来求下面的积分：

$$P_1=\int\frac{\sin t}{t^2+1}\mathrm{d}t\ ,\quad P_2=\int_0^{\pi}\frac{\sin t}{t}\mathrm{d}t$$

分析： 在例 13-4 中的两个积分分别是不定积分和定积分。对于定积分的求解，微分方程的初值可以任意选择而不影响计算结果。其中的微分方程是一阶微分方程，它们为：

$F_1'(x)=\sin x/(x^2+1)$ ，$F_2'(x)=e^{\cos x}$

计算程序如下：

```
F1 = dsolve('DF1 = sin(t)/(t^2+1)');       % 求解微分方程
P1 = F1
F2 = dsolve('DF2 = sin(t)/t','F2(0)=0');    % 求解微分方程
P2 = subs(F2,pi)-subs(F2,0)               % 带入积分限求差
```

输出结果为：

```
P1 =
-1/2*i*sinint(t-i)*cosh(1)+1/2*cosint(t-i)*sinh(1)+1/2*i*sinint(t+i)*cosh(1)+1
/2*cosint(t+i)*sinh(1)+C1
P2 =   1.8519
```



过程结果 F1 和 F2 是符号型变量，而在不定积分中也可以得到一个积分常数 C1。

前面介绍了符号法求解积分问题，下面介绍 MATLAB 提供的函数计算数值积分。

对于变化缓慢的被积函数，使用等间距离散采样步长可以保证整体计算的精度。然而如果被积函数在某个范围内变化很剧烈，则为了保证计算精度就需要使用更小的采样步长。因此如果设计一种算法能在被积函数变化范围比较大的时候采样点密集，而在函数值变化缓慢的区间采用较稀疏的采样点，就可以在使用较少的采样点数得到很高的求解精度。这种算法也相应地被称为自适应积分法。下面介绍一些实现自适应积分的 MATLAB 函数。

13.2.1.2 函数 quad

MATLAB 中的函数 quad 采用自适应辛普森（Simpson）积分公式来计算数值积分问题，该函数可以用来计算一元定积分问题，其调用格式为：

```
q = quad(fun,a,b)
q = quad(fun,a,b,tol);
q = quad(fun,a,b,tol,trace);
[q,fcnt] = quad(fun,a,b,tol,trace);
```

参数说明： q 是计算的积分结果。fcnt 是被积函数计算的次数。fun 是函数的句柄。a 和 b 分别是积分的下限和上限，对于 a 和 b 的大小关系没有限制，如果用户交换 a 和 b 位置，所得结果是前面加一个负号。tol 是精度控制量，其为一个较小的数，默认值是 1e-6。参数 trace 用于在迭代过程中表示向量[fcnt,a,b-a,q]。其中输入参数 fun，a 和 b 是必需的。下面举例来说明该函数的用法。

例 13–6：计算下面的积分：

$$f_1 = \int_0^2 \sin 3x e^{-4x} \mathrm{d}x\ ,\quad f_2 = \int_0^2 \sin\left(\sqrt{x} - x^2\right)\mathrm{d}x\ ,\quad f_3 = \int_0^2 \frac{1+\cos x}{c+e^x}\mathrm{d}x\ ,\quad f_4 = \int_0^2 \cos\left(\frac{2}{x+d}\right)\mathrm{d}x$$

分析：被积函数可以有两种方法来定义，即使用一个函数文件来定义和使用函数 inline 来定义。在第 3 个和第 4 个积分中还存在着常数 c 和 d。下面给出的程序分别使用这两种方法定义，需要注意的是函数表达式中的乘、除、乘方等运算需要支持向量计算，否则要出现错误提示。

下面两个函数文件 intfun1.m 和 intfun2.m 的内容是分别用来定义第 1 个和第 4 个积分表达式中的被积函数的。

```
function y = intfun1(x);
y = sin(3*x).*exp(-4*x); % 被积函数表达式

function y = intfun2(x,d);
y = cos(2./[x+d]); % 被积函数表达式
```

第 2 个和第 3 个被积函数在主文件中定义，主程序内容为：

```
c=2; % 参数赋值
d=3; % 参数赋值
fun2 = inline('sin(sqrt(x)-x.^2)');              % 定义第 2 个积分中的被积函数
fun3 = inline('[1+cos(x)]./[c+exp(x)]','x','c');  % 定义第 3 个积分中的被积函数
f1 = quad('intfun1',0,2)
f2 = quad(@(x)fun2(x),0,2,1e-5)
f3 = quad(@(x)fun3(x,c),0,2,1e-5)
f4 = quad(@(x)intfun2(x,d),0,2)
```

输出结果为：

```
f1 =    0.1200
f2 =   -0.3331
f3 =    0.6837
f4 =    1.7397
```



用户可以用引号和符号@来标识函数句柄，但是需要注意它们的用法和使用范围。引号方式适用于函数文件定义的被积函数。符号@适用于两种定义被积函数，但应注意需要使用下面的格式：@(x)funname(x)或者@(x)funname(x,c)。为了不发生错误，不熟练的用户可以都使用符号@来标识函数句柄。下面的语句是不正确的，会出现错误提示信息。

```
f2p = quad(@fun2,0,2)
f2q = quad('fun2',0,2)
```

13.2.1.3 函数 quadv

函数 quadv 和函数 quad 的功能类似，不同的是 quadv 可以求解被积函数中含向量参数的情况。函数 quadv 的调用格式为：

```
q = quadv(fun,a,b);
q = quadv(fun,a,b,tol);
q = quadv(fun,a,b,tol,trace);
[q,fcnt] = quadv(fun,a,b,tol,trace);
```

参数说明：该函数的输入和输出参数与函数 quad 的参数含义和用法类似，不同之处是对于函数 quadv，被积函数 fun 中还有向量或者是由多个表达式组成的向量函数，因此输出积分结果 q 的行数和列数与输入参数 fun 的行数和列数相同。下面举例来说明该函数的用法。

例 13-7：求解下面的积分，其中参数 v 是一个向量，其值为 $v=[1,2,3,4,5]$。

$$u_1=\int_0^1 x^3\cos(vx)\mathrm{d}x\text{ , }u_2=\int_0^1[x^2\cos(2x),x^3\sin(x)]\mathrm{d}x\text{ ,}$$

$$u_3=\int_0^1\begin{bmatrix}x\cos x & x^2\sin x\\ x^2\cos x & x\cos 2x\end{bmatrix}\mathrm{d}x$$

分析：这两个积分中，第 1 个积分含有一个向量参数，第 2 个积分被积函数是由两个函数组成的向量函数，第 3 个积分被积函数是一个矩阵形式。这里给出一种直接输入被积函数的形式，即不用编写函数文件或者调用 inline 函数来定义积分函数。相应的计算程序如下：

```
u1 = quadv(@(x)x.^3.*cos([1:5]*x),0,1) % 计算积分u1，参数 v 以向量形式输入
u2 = quadv(@(x)[x.^2.*cos(2*x),x.^3.*sin(x)],0,1) % 计算积分u2
u3 = quadv(@(x)[x.*cos(x),x.^2.*sin(x);x.^2.*sin(x),x.*cos(2*x)],0,1) % 计算积分
u3，4 个函数以矩阵形式输入
```

输出为：

```
u1 =    0.1717   -0.0084   -0.1669   -0.2021   -0.1048
u2 =    0.0193    0.1771
u2 =    0.3818    0.2232
       0.2232    0.1006
```

可见利用函数 quadv 可以同时计算多个积分，这一点在实际应用中是非常便利的，若要用函数 quad，则需要使用循环结构计算。

13.2.1.4 函数 quadl

函数 quadl 采用自适应洛巴托（Lobatto）积分法来计算数值积分，其调用格式为：

```
q = quadl(fun,a,b);
q = quadl(fun,a,b,tol);
q = quadl(fun,a,b,tol,trace);
[q,fcnt] = quadl(fun,a,b,tol,trace);
```

参数说明：从上面介绍的调用格式可以发现函数 quadl 和函数 quad 完全相同，它们的输入、输出参数的意义也相同，这里不再赘述。函数 quad 一般对于不光滑函数较低精度有效。而函数 quadl 对于光滑函数较高精度有效。下面举例来说明这个函数的用法。

例 13-8：求下面函数的积分，其中参数 $a=2.3$，$b=1.5$。

$$q_1=\int_0^2\frac{\mathrm{d}x}{(x-2)^2+5}\text{ , }q_2=\int_0^2\frac{\mathrm{d}x}{(x+2)^a+b}$$

分析： 这个问题中，两个被积函数都是分式形式，而第 2 个积分中含有两个参数。这里分别考虑使用两种方式来定义被积函数。

利用下面的函数文件 intfun3.m 来定义积分 q_2 中的被积函数，内容如下：

```
function y = intfun3(x,a,b);
y=(x+2).^a+b;                    % 被积函数表达式
```

第 1 个被积函数使用 MATLAB 函数 inline 来定义，同时写入主程序。主程序内容如下：

```
fun1 = inline('(x-2).^2+5');
tol = 1e-8;        % 定义积分精度
a=2.3;             % 对参数 a 赋值
b=1.5;             % 对参数 b 赋值
q1 = quadl(@(x)fun1(x),0,2)
q2 = quadl(@(x)intfun3(x,a,b),0,2)
```

输出结果为：

```
q1 =   12.6667
q2 =   29.4111
```

这里需要说明的是函数 quad 和 quadl 的积分限不能包含无穷大，比如下面的例子：

```
ff = inline('1./x.^2');             % 定义被积函数
q3 = quad(@(x)ff(x),1,inf)      % 调用函数 quad 计算从 1 到无穷的积分
q4 = quadl(@(x)ff(x),1,inf)      % 调用函数 quadl 计算从 1 到无穷的积分
```

输出为：

```
Warning: Infinite or Not-a-Number function value encountered.
> In quad at 109
  In example13_6 at 11
q3 =   NaN
Warning: Minimum step size reached; singularity possible.
> In quadl at 101
  In example13_6 at 12
q4 =   Inf
```

众所周知，$\int_1^{\infty}\frac{1}{x^2}\mathrm{d}x=1$，显然上面两个计算积分的函数 quad 和 quadl 对于这样的积分是无能为力的。

13.2.1.5　函数 quadgk

函数 quadgk 采用自适应高斯-克朗罗德（Gauss-Kronrod）积分法来计算数值积分，该函数可以用来解决含有无穷区间端点的积分、端点中等奇异的积分以及沿分段线性路径的路径积分。其调用格式为：

```
q = quadgk(fun,a,b);
[q,errbnd] = quadgk(fun,a,b,param1,val1,param2,val2,...);
```

参数说明： q 是输出的结果。errbnd 是一个绝对误差的近似范围。fun 是被积函数对应的句柄。a 和 b 是积分的上下限。param1 和 param2 表示属性名。val1 和 val2 是属性相应的取值。其中的

属性名包括 AbsTol 是绝对误差范围，其默认值是 1e-10；RelTol 是相对误差范围，其默认值是 1e-6，输出参数 errbnd 不大于 max(AbsTol,RelTol*|q|)；Waypoints 是积分区间内所有中断点按单调递增或者递减顺序组成的一个向量，其中奇异点不能包含在 Waypoint 向量里面，奇异点只能是区间端点；MaxIntervalCount 是允许区间的最大数目，其默认值是 650，超过这个数值 MATLAB 将会以警告的方式通知用户。下面举例说明函数 quadgk 的用法。

例 13-9： 计算下列积分的数值。

$$w_1=\int_1^{\infty}\frac{\mathrm{d}x}{x^2}\,,\quad w_2=\int_0^4 p(x)\,\mathrm{d}x\,,\quad w_3=\int_s\frac{1}{2z-1}\mathrm{d}z$$

其中，函数 $p(x)$ 和参数 s 如下定义：

$$p(x)=\begin{cases}x, & x\in[0,1]\\ \sin x, & x\in(1,2)\\ \cos x, & x\in[2,3)\\ x^2-3x, & x\in[3,4]\end{cases},\qquad s:\ 0\rightarrow 1+\mathrm{i}\rightarrow 1-\mathrm{i}\rightarrow 0$$

分析： 这是 3 种不同类型的积分，即积分 w_1 的上积分限是无穷大，积分 w_2 是一个分段不连续函数，积分 w_3 是一个路径积分。下面调用函数 quadgk 来求解这 3 个积分的数值。

这里使用函数文件 intfun4.m 来定义 w_2 中的被积函数，具体如下：

```
function y = intfun4(x);
y = x.^2-3*x;
y(x>=0&x<=1) = x(x>=0&x<=1);     % 被积函数表达式
y(x>1&x<2) = sin(x(x>1&x<2));     % 被积函数表达式
y(x>=2&x<3) = cos(x(x>=2&x<3));  % 被积函数表达式
```

其中针对不同区间的取值进行了计算，此外用户还可以使用 if 语句来定义这个分段函数。其他两个函数在主程序中定义，主程序内容如下：

```
fun1 = inline('1./x.^2');                          % 定义第一个积分中的被积函数

w1 = quadgk(@(x)fun1(x),1,inf)                      % 求第一个积分
format long
w2 = quadgk(@(x)intfun4(x),0,4,'AbsTol',1e-3)            % 求第二个积分
w2p = quadgk(@(x)intfun4(x),0,4,'Waypoints',[1,2,3],'AbsTol',1e-3)  % 求第二个积分
w2q = quad(@(x)intfun4(x),0,4)                        % 求第二个积分
Q = quadgk(@(z)1./(2*z-1),0,0,'Waypoints',[1+i,1-i])         % 求第三个积分
format short
```

输出结果如下：

```
w1 =    1
w2 =   2.521890211276745
w2p =   2.521605056982800
w2q =   2.521598038417038
Q=-0.000000000000000-3.141592653589800i
```

可见函数 quadgk 可以用来成功地求解一些特殊的积分问题：对于无穷积分限的情况，该函数可以得到正确的结果；对于含有断点的函数可以通过设置断点的属性来改善结果，比较上面的结果

w2p 和 w2q 可以看出，设置断点位置（w2q）和不设置（w2p）得到的结果在较小的小数位上不一样，其中 w2q 更精确些，并且随着绝对精度参数 AbsTol 基本不变；对于第 3 个积分的计算语句，用户可以使用 “@(z)1./(2*z-1)” 形式来直接输入被积函数，这样对于解决表达式简单的积分问题十分方便，使得编程简单。

此外对于断点奇异的积分，函数 quadgk 可以解决，比如计算下面的积分：

$$w_4 = \int_0^{\infty} e^{-x^2} (\ln x)^2 \, dx$$

被积函数在 $x = 0$ 处是奇异的。这样的积分利用前面的函数 quad 和 quadl 是无法计算的，而函数 quadgk 可以轻松搞定。相应的计算语句如下：

```
w4 = quadgk(@(x)exp(-x.^2).*log(x).^2,0,inf)
```

输出结果为：

```
w4 =    1.9475
```

通过上面的介绍，可以发现函数 quadgk 可以有效地解决一些不易计算的积分，希望用户熟练掌握这个函数的用法，来解决实际应用中的积分难题。

13.2.1.6　函数 dblquad

函数 dblquad 可以用来计算二重积分问题，其调用格式为：

```
q = dblquad(fun,xmin,xmax,ymin,ymax);
q = dblquad(fun,xmin,xmax,ymin,ymax,tol)
q = dblquad(fun,xmin,xmax,ymin,ymax,tol,@quadl);
q = dblquad(fun,xmin,xmax,ymin,ymax,tol,myquadf);
```

参数说明： q 是输出的积分结果。fun 表示被积函数。xmin 和 xmax 分别对应于变量 x 的下积分限和上积分限。ymin 和 ymax 分别对应于变量 y 的下积分限和上积分限。tol 是精度控制量。@quadl 表示使用求积分函数 quadl 来代替默认的 quad 计算。myquadf 表示用户自定义积分函数来代替函数 quad。下面举例说明函数 dblquad 的用法。

例 13-10： 求下面的二重积分，其中参数 C 表示圆 $x^2 + y^2 = 1$ 内部的区域，E 表示椭圆 $x^2/2 + y^2/3 = 1$ 内部的区域。

$$K_1 = \int_0^1\int_0^2 x\sin y - y\cos x \, dx dy, \quad K_2 = \int_0^1\int_0^3 xe^y \, dx dy, \quad K_3 = \int_0^{\pi}\int_0^2 \sin(x+y) \, dx dy,$$

$$K_4 = \int_0^2\int_0^1 \frac{2xy}{1+x^2+y^2} \, dx dy, \quad K_5 = \iint_C \cos(2x+y) \, dx dy, \quad K_6 = \iint_E \sin(x^2 - 2y) \, dx dy$$

分析： 问题中含有 6 个积分，其中前 4 个的积分区域形状是规则的矩形，而后两个积分的区域形状是非矩形，用户可以把积分区域外部的函数值设置为 0，即可按包含积分区域的最小矩形求积分。这个技巧可以扩展到求解其他不规则区域的二重积分计算。下面来建立求解程序。

为了简便，这里直接输入被积函数而不另行建立函数文件或者调用 inline 函数来定义，相应的计算程序如下：

```
tol = 1e-5;                    % 精度控制
myquadf = inline('quadgk');    % 定义积分函数
K1 = dblquad(@(x,y)x*sin(y)-y*cos(x),0,2,0,1)
K2 = dblquad(@(x,y)x*exp(y),0,3,0,1,tol)
K3 = dblquad(@(x,y)sin(x+y),0,2,0,pi,'quadl')
K4 = dblquad(@(x,y)2.*x.*y./[1+x.^2+y.^2],0,2,0,pi,'myquadf')
K5 = dblquad(@(x,y)cos(2*x+y).*sign(sqrt(max(1-(x.^2+y.^2),0))),-1,1,-1,1)
K6 =
dblquad(@(x,y)sin(x.^2-2*y).*(x.^2/2+y.^2/3<=1),-sqrt(2),sqrt(2),-sqrt(3),sqrt
(3))
```

输出结果为：

```
K1 =    0.4647
K2 =    7.7323
K3 =    1.8186
K4 =    3.0781
K5 =    1.5471
K6 =    0.8129
```

对于 myquadf，用户可以调用 quadgk 函数，其利用函数 inline 可以简单地定义，同时 tol 后面积分函数的标识需要使用引号，若使用符号@会出现错误提示；圆形的积分区域利用这个语句段 sign(sqrt(max(1-(x.^2+y.^2),0)))来实现，其原理是：首先比较函数 $f(x,y)=1-x^2-y^2$ 和 0 的大小关系，区域内部的值大于 0，外部的值小于 0，而边界的值等于 0，因此通过和 0 比较大小关系可以提取出问题中的积分区域，再通过符号函数可以把内部的正数变为 1，从而不影响积分结果；椭圆积分区域通过简单的语句(x.^2/2+y.^2/3<=1)来提取。这个方法可以进一步推广，对于某些封闭曲线（ $f(x,y)=0$ ）内部的区域可以通过 $f(x,y)>0$ 提取，比如第 5 个积分中的积分区域可以通过(x.^2+y.^2<=1)来实现。

13.2.1.7 函数 triplequad

函数 triplequad 可以用来计算三重积分问题，其调用格式为：

```
q = triplequad(fun,xmin,xmax,ymin,ymax,zmin,zmax);
q = triplequad(fun,xmin,xmax,ymin,ymax,zmin,zmax,tol);
q = triplequad(fun,xmin,xmax,ymin,ymax,zmin,zmax,tol,@quadl);
q = triplequad(fun,xmin,xmax,ymin,ymax,zmin,zmax,tol,myquadf);
```

参数说明：函数 triplequad 的调用格式和函数 dblquad 相似。q 是输出的积分结果。fun 表示被积函数。xmin 和 xmax 分别对应于变量 x 的下积分限和上积分限，ymin 和 ymax 分别对应于变量 y 的下积分限和上积分限，zmin 和 zmax 分别对应于变量 z 的下积分限和上积分限。tol 是精度控制量，其默认值是 1e-6。@quadl 表示使用求积分函数 quadl 代替默认的 quad 来计算。myquadf 表示用户自定义积分函数来代替函数 quad。下面举例说明函数 triplequad 的用法。

例 13-11：求下面三重积分的数值，其中参数 E 表示椭圆球 $x^2/2+y^2/3+z^2/4=1$ 内部区域。

$$L_1=\int_0^1\int_0^2\int_0^3 x\sin(y+z)\,\mathrm{d}x\mathrm{d}y\mathrm{d}z\text{ , }L_2=\int_0^1\int_0^2\int_0^3\frac{x+y+z}{x^3+y^3+z^3+1}\mathrm{d}x\mathrm{d}y\mathrm{d}z\text{ ,}$$

$$L_3 = \iiint\limits_E \sin x^2 + z^2 \cos y \mathrm{d}x\mathrm{d}y\mathrm{d}z$$

分析：对于三重积分的计算可以进行与二重积分类似的处理，前两个积分的积分区域是长方体，而最后一个积分的积分区域是非矩形区域，可考虑使用类似于例 13-10 中第 6 个积分那样的处理方法，通过增加一个调制项转化为对被积函数的调整。相应的求解语句如下所示。

```
tol = 1e-5;           % 精度控制
L1 = triplequad(@(x,y,z)x*sin(y+z),0,3,0,2,0,1) % 定义区间计算三重积分
L2 = triplequad(@(x,y,z)[x+y+z]./[x.^2+y.^2+z.^2+1],0,3,0,2,0,1,tol,'quadl')
L3 = triplequad(@(x,y,z)[sin(x.^2)+z^2*cos(y)].*(x.^2/2+y.^2/3+z.^2/4<=1),
-sqrt(2),sqrt(2),-sqrt(3),sqrt(3),-2,2)
```

输出结果是：

```
L1 =    7.2434
L2 =    3.4990
L3 =   20.1933
```

通过例 13-10 和例 13-11 可以发现，利用 MATLAB 计算一些多重积分是非常方便的，同时也可以进行一些特殊积分区域问题的计算。

13.2.2 累加法

本小节介绍经典的数值积分算法，如梯形积分、龙贝格（Romberg）积分、辛普森积分以及逐步区间二分法等。通过这些积分算法的介绍，用户可以更深刻地理解数值积分思想。

13.2.2.1 梯形积分法

首先考虑在分割的子区间$[x_k, x_{k+1}]$内，其积分值可以有如下 3 种方式来计算。

◆ 利用中点的函数值代替整个区间的函数值：

$$\int_{x_k}^{x_{k+1}} f(x)\mathrm{d}x \approx (x_{k+1} - x_k) f\left(\frac{x_k + x_{k+1}}{2}\right) \tag{13-6}$$

◆ 利用梯形面积公式来计算积分值：

$$\int_{x_k}^{x_{k+1}} f(x)\mathrm{d}x \approx \frac{x_{k+1} - x_k}{2}\left[f(x_k) + f(x_{k+1})\right] \tag{13-7}$$

◆ 在式（13-6）的基础上，在子区间再细分一步的结果为：

$$\int_{x_k}^{x_{k+1}} f(x)\mathrm{d}x \approx \frac{x_{k+1} - x_k}{6}\left[f(x_k) + 4f\left(\frac{x_k + x_{k+1}}{2}\right) + f(x_{k+1})\right] \tag{13-8}$$

利用上面 3 种公式可以得到不同的数值积分算法，结果的精度随着分割步长的减小而提高。梯形积分法的计算公式如下：

$$\int_a^b f(x)\mathrm{d}x = \sum_{k=0}^{n-1}\int_{x_k}^{x_{k+1}} f(x)\mathrm{d}x = h\left[\frac{f(a) + f(b)}{2} + \sum_{k=1}^{n-1} f(x_k)\right] \tag{13-9}$$

其中 n 表示在区间$[a,b]$上等分子区间的数目。h 为等分区间的步长。上述公式可以从式（13-7）获得，感兴趣的用户可以推导一下。梯形积分法误差的阶是$O(h^2)$。根据公式（13-9）可

以写出梯形积分法的程序，具体内容如下：

```
function y = trapez(fun,a,b,n,varargin);
if nargin<3;
    error('fun,a,b must be defined');     % 输入参数少于 3 个提示出错
elseif nargin==3;
    n = 1000;                             % 设置 n 的默认值
end
h = [b-a]/n;                              % 计算步长
x = a+h*[0:n];                            % 计算采样点
if isempty(varargin);
    fx = feval(fun,x);                    % 计算函数值
else
    fx = feval(fun,x,varargin{:});        % 计算函数值
end
y = h*[(fx(1)+fx(end))/2+sum(fx(2:end-1))];  % 按公式算积分
```

参数说明： y 是输出的积分计算结果。fun 是被积函数的句柄。a 和 b 分别是积分的上下限。n 是等间距分割区间的数目，其默认值为 1000。varargin 表示 fun 函数的不定数目的参数。特别注意，上面程序中最后一行语句可以使用一个等价形式替换：y = h*trapz(fx)。下面举例说明这个函数的用法。

例 13–12： 利用梯形积分法计算下面的积分，其中参数 $c=3$。

$$Q_1=\int_0^2 x\sin x\mathrm{d}x\ ,\quad Q_2=\int_0^1 x\sin(cx+2)\mathrm{d}x$$

分析： 为了保证程序的独立性，这里使用函数 inline 来定义被积函数。参数 c 使用传递的方式输入到前面定义的梯形法数值积分程序中。使用不同的调用方式来计算这两个积分。

程序如下：

```
c=3;                                   % 参数赋值
f1 = inline('x.*sin(x)');              % 定义被积函数
Q1 = trapez(@(x)f1(x),0,2)    % 引用函数 f1
Q2 = trapez(@(x,c)x.*sin(c*x+2),0,1,2000,c)    % 此处直接输入被积函数
```

输出的计算结果为：

```
Q1 =    1.7416
Q2 =   -0.3021
```

从这个例子可看出，用户可编写函数如同前面介绍的函数 quad 一样来计算积分问题。

13.2.2.2 龙贝格积分法

接下来介绍龙贝格积分法，该算法计算函数 $f(x)$ 在区间 $[a,b]$ 上定积分的步骤如下：

step 1 初始化参数 $k=1$，并计算最粗略的积分。

$$r_{1,1}=\frac{b-a}{2}\left[f(a)+f(b)\right] \tag{13-10}$$

step 2 计算下一个位置的值。

$$r_{k+1,1}=\frac{1}{2}\left[r_{k,1}+\frac{b-a}{2^{k-1}}\sum_{n=1}^{M}f\left(a+(2n-1)\frac{b-a}{2^k}\right)\right],\quad M=2^{k-1} \tag{13-11}$$

step 3 利用下面的递推公式计算。

$$r_{k+1,m}=\frac{4^{m-1}r_{k+1,m-1}-r_{k,m-1}}{4^{m-1}-1},\quad m=2,3,\ldots,k+1 \tag{13-12}$$

step 4 如果 $r_{k+1,k+1}$ 和 $r_{k,k}$ 之间的差值满足预定义的精度，则终止计算，否则令 $k\to k+1$，进入到步骤 2 不计算。

龙贝格积分法的过程将组成下面的下三角矩阵：

$$R=\begin{bmatrix} r_{1,1} \\ r_{2,1} & r_{2,2} \\ r_{3,1} & r_{3,2} & r_{3,3} \\ r_{4,1} & r_{4,2} & r_{4,3} & r_{4,4} \\ r_{5,1} & r_{5,2} & r_{5,3} & r_{5,4} & r_{5,5} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{13-13}$$

龙贝格积分法的思想可以理解为不断对分分割步长，并累加增加的采样点函数值。下面给出龙贝格积分法的实现程序：

```
function y = romberg(fun,a,b,tol,varargin);
if nargin<3;
   error('fun,a,b must be defined');  % 输入参数少于 3 个提示出错
elseif nargin==3;
   tol = 1e-4;                        % 设置 tol 的默认值
end
fab = feval(fun,[a,b],varargin{:});
r(1,1) = [b-a]/2*sum(fab);
k = 0;                 % 初始化脚标参数
D = inf;                % 设置误差的初值
while D>tol;
   k = k+1;            % 更新脚标参数
   x = a+(b-a)/2^k*[2*[1:2^(k-1)]-1];      % 计算采样点坐标
   fx = feval(fun,x,varargin{:});          % 计算函数值
   r(k+1,1)=0.5*[r(k,1)+(b-a)/2^(k-1)*sum(fx)];        % 计算第一列中的下一行元素值
   for m=2:k+1;
      r(k+1,m)=[4^(m-1)*r(k+1,m-1)-r(k,m-1)]/[4^(m-1)-1];    % 计算下一行元素值
   end
   D = abs(r(k+1,k+1)-r(k,k));      % 更新当前误差
end
y=r(k+1,k+1);                   % 返回积分值
```

参数说明： y 是输出的积分计算结果。fun 是被积函数的句柄。a 和 b 分别是积分的上、下限。tol 是误差的控制参数，其默认值为 1e-4。varargin 表示 fun 函数的不定数目的参数。下面举例说明这个函数的用法。

例 13-13： 利用龙贝格积分法计算下面的两个积分，其中系数的数值为 $b=5$，$c=4$。

$$V_1=\int_0^{\pi}\sin x\mathrm{d}x,\quad V_2=\int_0^1\frac{2x+b}{1+cx^2}\mathrm{d}x$$

分析： 这个问题中第一个积分是一个简单的积分，其积分值可以解析获得，即 $V_1=2$。在第二

个积分中含有两个系数，它们可以利用类似例 13-11 的方法输入到龙贝格积分函数中。下面考虑问题的求解。求解的程序如下：

```
b=5;     % 参数赋值
c=4;     % 参数赋值
tol=1e-6;  % 误差量
V1 = romberg(@(x)sin(x),0,pi)
V2 = romberg(@(x,b,c)[2*x+b]./[1+c*x.^2],0,1,tol,b,c)
```

输出结果为：

```
V1 =    2.0000
V2 =    3.1702
```

可以发现，龙贝格积分法需要计算很多过程参数的值，因此其计算量大且收敛速度慢，但是利用龙贝格计算法可以得到较高的计算精度。

13.2.2.3 辛普森积分法

辛普森积分法的计算公式可以按下式描述进行：

$$\int_a^b f(x)\mathrm{d}x = \frac{h}{3}\left[\frac{f(a)+f(b)}{2}+\sum_{k=1}^{n-1} f(x_k)\right]+\frac{2h}{3}\sum_{k=0}^{n-1} f\left(\frac{x_k+x_{k+1}}{2}\right) \quad (13\text{-}14)$$

其中 n 等分积分区间的采样坐标为 $\{x_0,x_1,\dots,x_n\}$，$x_0=a$，$x_n=b$。在公式（13-14）中，右侧的第一项与公式（13-9）有着密切的联系。辛普森积分法的误差余项为 $O(h^4)$。

根据公式（13-14）可以写出辛普森积分法的实现程序 simpson.m，具体内容如下：

```
function y = simpson(fun,a,b,n,varargin);
if nargin<3;
   error('fun,a,b must be defined');       % 输入参数少于 3 个提示出错
elseif nargin==3;
   n = 1000;                        % 设置 n 的默认值
end
h = [b-a]/n;                          % 计算步长
x = a+h*[0:n];                        % 计算采样点
fx = feval(fun,x,varargin{:});              % 计算函数值
y = h/3*trapz(fx,2);                   % 积分值的第一部分
fx = feval(fun,[x(1:end-1)+x(2:end)]/2,varargin{:});      % 计算分割区间中点处的函数
值
y = y+2*h/3*sum(fx,2);                        % 积分值的第二部分
```

参数说明： y 是输出的积分计算结果。fun 是被积函数的句柄。a 和 b 分别是积分的上、下限。n 是等间距分割区间的数目，其默认值为 1000。varargin 表示 fun 函数的不定数目的参数。特别地，上面程序还支持多个函数输入的情况，此时要求输入函数是按列向量顺序排列的。下面举例说明这个函数的用法。

例 13-14： 利用辛普森积分法计算下面的积分。

$$U_1=\int_0^1 e^x\sin x\mathrm{d}x\text{，}\quad U_2=\int_0^1\left[\sin ax,\cos x^b,e^{x+c},\frac{1}{x^2+f}\right]\mathrm{d}x$$

参数取值为 $a=1$，$b=2$，$c=3$，$f=4$。

分析：第一个积分是一个标准的积分，用户按函数 simpson 输入参数顺序进行计算即可。第二个积分含有参数的同时还是一个向量组成的被积函数，用户只要把 4 个被积函数排为列向量顺序即可。实现程序如下：

```
a=1.5; % 参数赋值
b=2; % 参数赋值
c=3; % 参数赋值
f=4; % 参数赋值
U1 = simpson(@(x)exp(x).*sin(x),0,1)   % 计算第一个积分 U1
U2 =
simpson(@(x,a,b,c,f)[sin(x*a);cos(x.^b);exp(x+c);1./[x.^2+f]],0,1,2000,a,b,c,f
); % 计算 U2
U2 = U2' % 转置显示
```

输出结果为：

```
U1 =    0.9093
U2 =    0.6195    0.9045   34.5126    0.2318
```



可见支持同一区间多个函数计算积分的功能对于计算多个函数的数值积分功能非常方便，感兴趣的用户可以把前面的函数 trapez 和 romberg 改为支持多函数输入的格式，改动时请参考函数 simpson。

13.2.2.4 逐步区间二分法

前面介绍的梯形积分法和辛普森积分法的计算公式非常简单，但是对于积分区间$[a,b]$分割的子区间数目用户在计算之前需要根据误差余项来估算，这一点在实际应用中不是很方便。下面来介绍逐次区间二分法。其基本思想就是不断地二分分割子区间，直至达到预先定义的精度而终止程序计算。其迭代公式可表示为：

$$D_{2n}=\frac{1}{2}D_n+h_{2n}\sum_{k=1}^{n}f\left(a+\left(2k-1\right)h_{2n}\right) \tag{13-15}$$

其中n是分割的区间数目。D_n是二分区间的积分值。h_{2n}是二分后的步长，其值为$h_{2n}=h_n/2$。计算的误差可以通过检验不等式$|D_{2n}-D_n|<\varepsilon$是否成立来控制，其中ε是一个较小的正数，即绝对误差。对于初始分割区间数目，用户可以选择任意正整数。这里使用$n=1$作为初值，初始积分值为$\left[f(a)+f(b)\right]/2$。该积分法相应的计算程序如下：

```
function y = divide2(fun,a,b,tol,varargin);
if nargin<3;
   error('fun,a,b must be defined');   % 输入参数少于 3 个提示出错
elseif nargin==3;
   tol = 1e-4;                         % 设置 tol 的默认值
end
y0 = sum(fun([a,b],varargin{:}),2)/2;   % 初始化过程量的数值
y = y0;                                % 初始化积分值
Er = inf;                              % 初始误差
h = b-a;                               % 初始步长
n=1;                                   % 初始区间数
while Er>tol;
   y0 = y;                             % 更新 y0 的数值
```

```
    h = h/2;                         % 二分区间长度
    n=n*2;                           % 双倍区间数目
    x = a+(2*[1:n/2]-1)*h;           % 计算采样点
    y = y0/2+h*sum(fun(x,varargin{:}),2);  % 计算函数值同时更新积分值
    Er=max(abs(y-y0));               % 计算当前误差
End
```

参数说明： y 是输出的积分计算结果。fun 是被积函数的句柄。a 和 b 分别是积分的上、下限。tol 是绝对误差，其默认值 1e-4。varargin 表示 fun 函数的不定数目的参数。特别地，上面程序支持多个函数输入的情况，此时要求输入函数按列向量顺序排列。下面举例说明这个函数的用法。

例 13–15： 使用上面的函数 divide2 计算下面两个积分：

$A_1=\int_{-1}^{1}e^x\cos x\mathrm{d}x$，$A_2=\int_{-1}^{1}\left[x\sin ax,\cos x^b,xe^{x/c}\right]\mathrm{d}x$，参数取值为 $a=1.5$，$b=2.2$，$c=\pi$。

分析： 第一个积分按函数 divide2 的一般格式计算即可。在第 2 个积分中被积函数是一个 1×3 的向量同时含有 3 个参数，把 3 个函数按列向量方式输入到 divide2 函数即可。计算程序如下：

```
a=1.5;        % 参数赋值
b=2.2;        % 参数赋值
c=pi;         % 参数赋值
U1 = divide2(@(x)exp(x).*cos(x),-1,1)      % 计算第一个积分 U1
U2 = divide2(@(x,a,b,c,f)[x.*sin(x*a);cos(x.^b);x.*exp(x/c);],0,1,1e-6,a,b,c);
% 计算第二个积分 U2
U2 = U2'     % 转置显示
```

输出结果为：

```
U1 =    1.9334
U2 =    0.3962    0.9116    0.6199
```

通过使用本节介绍的 MATLAB 函数以及后面给出的数值积分算法的实现函数，用户可以求解定义区间 $[a,b]$ 上连续函数或者分段连续函数，以及含有中等奇异端点的数值积分。此外，还存在一些利用特殊函数零点来计算数值积分的算法，比如高斯-勒让德积分法、高斯-切比雪夫积分法、高斯-拉盖尔积分法以及高斯-厄尔米特积分法等，这里不再介绍，感兴趣的用户可以查找相关文献编写相应程序。对于一些含有奇异点的积分问题将在下一节介绍。

13.3 奇异积分计算

在一些问题中，会出现奇异积分的情况。本节介绍这类积分的处理办法。

有的奇异积分可以通过变量替换的办法转为非奇异积分。比如：

$q_1=\int_0^1 x^{-\frac{1}{n}}f(x)\mathrm{d}x\xrightarrow{u^n=x}n\int_0^1 f(u^n)u^{n-2}\mathrm{d}u$，其中 $n\geq2$

有的奇异积分可以使用函数 quadgk 来计算，比如：

$q_2=\int_0^1\frac{\mathrm{d}x}{x^{\frac{1}{2}}+x^{\frac{1}{3}}}$，$q_3=\int_0^1\frac{e^x}{\sqrt{x}}\mathrm{d}x$，$q_4=\int_0^1\frac{1}{\sqrt{x(1-x)}}\mathrm{d}x$，$q_5=\int_{-1}^1\frac{\sin x}{x}\mathrm{d}x$

积分 q_2 和 q_3 在 $x=0$ 处是奇异的，可以使用如下语句计算：

```
q2 = quadgk(@(x)1./[x.^(1/2)+x.^(1/3)],0,1,'AbsTol',1e-4)
q3 = quadgk(@(x)exp(x)./[x.^(1/2)],0,1,'AbsTol',1e-4)
q4 = quadgk(@(x)1./sqrt(x.*(1-x)),0,1,'AbsTol',1e-4)
q5 = quadgk(@(x)sin(x)./x,-1,1,'AbsTol',1e-4)
```

输出的结果为：

```
q2 =    0.8411
q3 =    2.9253
q4 =    3.1416
q5 =    1.8922
```

可见奇异点位于积分区间内部时，MATLAB 的函数 quadgk 也能给出正确结果，比如积分 q_5。如果利用函数 quadgk 不能给出结果或出现警告时，用户需要检查输入的被积函数是否有误、是否可积。

13.4 小结

本章主要介绍了积分问题的求解方法。首先介绍了利用函数 symsum 完成级数求和问题，并介绍了泰勒级数展开和傅里叶级数展开。接下来介绍了数值积分的计算方法，以及利用函数 int 进行符号积分的计算，利用求解微分方程的函数 dsolve 计算积分。函数 quad，quadl，quadgk，dblquad 以及 triplequad 计算数值积分的方法也在本章做了讲解。同时，本章还介绍了梯形积分法、龙贝格积分法、辛普森积分法、逐步区间二分法等数值积分算法的程序实现，最后介绍了利用函数 quadgk 计算含有奇异点的积分。利用本章介绍的知识，用户可以求解不同类型的积分问题。

◆ ◆ ◆

第 14 章　数学变换运算

本章包括

- ◆ **分数傅里叶变换**　介绍傅里叶变换的计算以及在此基础上计算分数傅里叶变换。
- ◆ **菲涅尔（Fresnel）变换**　介绍菲涅尔变换的数值计算方法。
- ◆ **Hartley 变换**　介绍 Hartley 变换的定义及计算程序。
- ◆ **离散正/余弦变换**　介绍 4 种形式的正/余弦变换的定义及其计算。
- ◆ **分数随机变换**　介绍分数随机变换的数学定义过程及程序实现。
- ◆ **汉克尔（Hankel）变换**　介绍汉克尔变换的定义及其数值算法。
- ◆ **小波变换**　介绍一维和二维小波变换函数。

本章主要介绍几种在基础研究和工程应用中用到的数学变换，它们是图像、信号处理问题中的重要工具。众所周知，傅里叶变换是信息处理的基本工具，广泛应用于各类问题中。然而对于有些特殊问题，傅里叶变换显得无能为力或者得不到理想的结果。正是这个原因，在 19 世纪 80 年代出现了分数傅里叶变换和小波变换，这两种变换弥补了傅里叶变换的不足。本章重点介绍除傅里叶变换以外的几种变换以及它们的实现程序。

14.1　分数傅里叶变换

在介绍分数傅里叶变换之前先简要回顾一下傅里叶变换。傅里叶变换的定义公式为：

$$X(u)=\int_{-\infty}^{\infty}x(t)e^{-i2\pi ft}\mathrm{d}t \tag{14-1}$$

其中 $X(u)$ 是傅里叶变换输出结果，u 是频域的坐标。$x(t)$ 是输入函数。$e^{-i2\pi ft}$ 是傅里叶变换的积分核。在 MATLAB 中，实现公式（14-1）对应的一维离散傅里叶变换的函数是 fft，而 fft2 和 fftn 可以分别用来实现二维和 n 维的傅里叶变换。这 3 个函数的调用格式如下：

```
X = fft(x);
X = fft(x,N);
X = fft(x,N,dim);
X = fft(x,[],dim);
X = fft2(x);
X = fft2(x,mrows,ncols);
X = fftn(x);
X = fftn(x,siz);
```

参数说明： X 是输出的变换结果。x 是输入的离散信号。N 用来控制输出信号的长度（或者为元素总数）。dim 用于在对矩阵进行一维变换时指定维数，为 1 表示对矩阵各列向量进行变换，为 2 表示对矩阵各行向量进行变换。mrows 和 ncols 分别用于指定输出的行数和列数。siz 用于定义输出数组各维度的元素数目。

例 14-1：对下面的函数进行一维傅里叶变换，其中 t 的取值范围是 $[-3,3]$，采样步长为 $\Delta t = 0.1$。

$$y(t) = \text{rect}(t) = \begin{cases} 1, & |t|<1 \\ 0, & |t|\geq 1 \end{cases}$$

分析：rect 代表矩形函数。首先离散采样获得函数采样点的函数值，然后根据采样步长计算出频域的坐标值：频域的区间是 $\left[-\frac{1}{2\Delta t}, \frac{1}{2\Delta t}\right]$，调用函数 fft 就可以进行离散傅里叶变换了。最后调用函数 fftshift 对输出结果进行位置移动，把零频移到中央。

首先生成函数 $y(t)$ 的离散形式，然后调用函数 fft 进行傅里叶变换，并给出相应变换结果的实部和虚部显示。这个过程的 MATLAB 程序如下：

```
t = -3:.1:3;                       % 得到离散采样点
dt = 1/2/[t(2)-t(1)];              % 计算采样步长
f = linspace(-dt,dt,length(t));    % 得到频域坐标点
y = zeros(size(t));                % 生成与 t 相同大小的全 0 向量
y(abs(t)<=1) = 1;                  % 得到离散的函数值
fy = fftshift(fft(y));             % 调用 fft 函数进行傅里叶变换并把零频移至中央
figure;subplot(131);plot(t,y);     % 绘制原信号曲线
title('signal','Fontsize',12);     % 注释图形内容
axis square;                       % 设置坐标轴为方形
subplot(132); plot(f,real(fy));    % 绘制输出的实数部分
title('real part','Fontsize',12);  % 注释图形内容
axis square; subplot(133); plot(f,imag(fy));  % 绘制输出的虚数部分
title('imaginary part','Fontsize',12);  % 注释图形内容
axis square;
```

输出结果如图 14.1 所示，左图给出的是输入信号，中间的图形是变换结果的实部，右图是虚部部分。

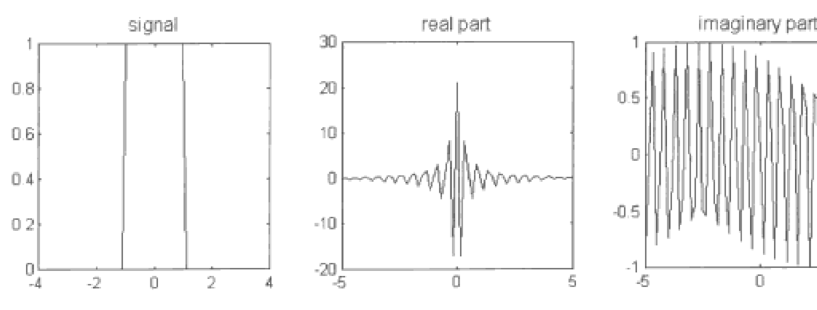


图 14.1 一维信号的离散傅里叶变换



用户还可以使用函数 abs 和 angle 分别得到输出的振幅和相角（也叫相位）。

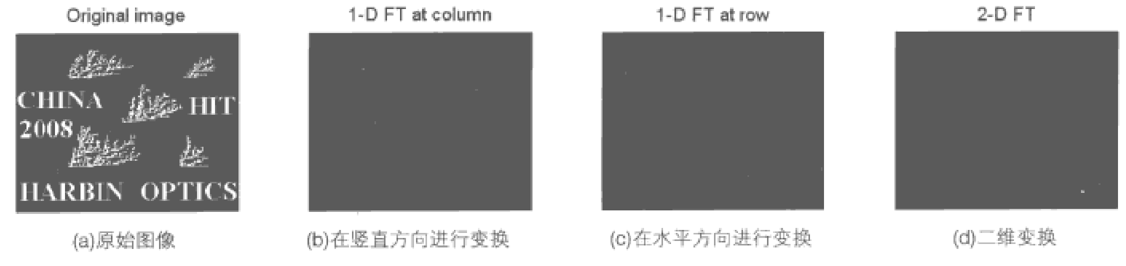
例 14-2：计算一幅图像的离散傅里叶变换，其中要求进行一维和二维变换。

分析：一幅图像在 MATLAB 中对应着一个矩阵或者一个 $M \times N \times 3$ 的数组，可以利用函数 imread 把图像的数据读入进行处理，从而调用函数 fft 或者 fft2 进行变换。这里同样需要调用函数 fftshift 来改变频谱的顺序，其中 fftshift(X,1)和 fftshift(X,2)分别对矩阵 X 的各个列向量和行向量进行位置移动。而 fftshift(X)用来对矩阵 X 进行位置移动，其对应于函数 fft2 的输出结果。

这里以光盘文件夹\Ch14 中 DLA.bmp 图片文件作为输入图像，读入图像之后再调用函数 fft2 进行傅里叶变换，同时给出利用一维变换函数 fft 进行变换作为比较。实现程序如下：

```
A = imread('DLA.bmp');        % 读入图像
A = double(A);               % 转换为 double 型数据
fa1 = fftshift(fft(A,[],1),1);    % 对列向量进行变换,并用 fftshit 函数移动频谱位置
fa2 = fftshift(fft(A,[],2),2);              % 对行向量进行变换，并用 fftshit 函数移动频
谱位置
fa = fftshift(fft2(A));                    % 进行二维变换
figure;subplot(141);imshow(abs(A),[]);     % 显示原图
title('Original image');                   % 在图题处注释
subplot(142);imshow(abs(fa1),[]);          % 逐列变换的结果
title('1-D FT at column');                 %  在图题处注释
subplot(143); imshow(abs(fa2),[]);          % 逐行变换的结果
title('1-D FT at row');                    % 在图题处注释
subplot(144); imshow(abs(fa),[]);           % 二维变换的结果
title('2-D FT');                          % 在图题处注释
set(gcf,'Color','w');                      % 设置背景色为白色
```

输出结果如图 14.2 所示，其中一维变换结果图(b)和图(c)主要在水平和竖直方向的中央位置处具有较高能量（白色），两侧位置能量较低（黑色），而图(d)的中心位置处有较高能量。



(a)原始图像　(b)在竖直方向进行变换　(c)在水平方向进行变换　(d)二维变换

图 14.2　图像傅里叶变换的结果



这里显示的是频谱中的振幅部分，相位部分的取值情况用户可以利用函数 angle 来计算。

傅里叶变换的逆变换实现的函数是 ifft，ifft2 和 ifftn，它们的用法与正变换相似，这里不再介绍。此外，还存在另外一种定义傅里叶变换的形式，即：

$$X(u)=\int_{-\infty}^{\infty}x(t)e^{-iut}\mathrm{d}t \tag{14-2}$$

在公式（14-2）中积分核缺少了系数 2π，这种形式常用于数学上的公式推导，在 MATLAB 中提供函数 fourier 来实现对应于式（14-2）变换的符号计算。该函数的调用格式为：

```
F = fourier(f);
F = fourier(f,v);
```

参数说明： F 是输出的傅里叶变换结果。f 是一个符号函数的表达式。V 在输出表达式中用来作为频率。

例 14-3： 下面是调用函数 fourier 对 $f_1=\exp\left(-x^2\right)$ 和 $f_2=x\exp\left(-y^2\right)$ 进行关于变量 x 的傅里叶

变换的例程。

```
syms x y v;
F1 = fourier(exp(-x^2))
F2 = fourier(exp(-y^2)/x,v)
```

输出结果为：

```
F1 =exp(-1/4*w^2)*pi^(1/2)
F2 =i*exp(-y^2)*pi*(1-2*heaviside(v))
```



其中函数 heaviside 是阶跃函数，用户可以使用 help heaviside 查看这个函数的信息。

分数傅里叶变换是傅里叶变换的一种推广形式。函数 $f(x)$ 的一维分数傅里叶变换的数学定义为：

$$\mathrm{F}^{\alpha}\{f(x)\}(u)=\int_{-\infty}^{+\infty} f(x)K_{\alpha}(x,u)\mathrm{d}x \tag{14-3}$$

其中：

$$K_{\alpha}(x,u)=\begin{cases} A_{\alpha}\exp\left[\mathrm{i}\pi\left(u^{2}\cot\theta_{\alpha}-2xu\csc\theta_{\alpha}+x^{2}\cot\theta_{\alpha}\right)\right], & \alpha\neq 2n \\ \delta(x+u), & \alpha=4n+2 \\ \delta(x-u), & \alpha=4n \end{cases} \tag{14-4}$$

$$A_{\alpha}=\sqrt{1-\mathrm{i}\cot\theta_{\alpha}}\,,\quad \theta_{\alpha}=\frac{\alpha\pi}{2} \tag{14-5}$$

上面的公式（14-3）至公式（14-5）给出了分数傅里叶变换的积分形式的定义，接下来再给出分数傅里叶变换的级数定义形式。函数 $f(x)$ 的分数傅里叶变换级数形式为：

$$F\alpha\{f(x)\}(u)=\sum_{n=0}^{\infty}C_{n}\exp(\mathrm{i}n\pi\alpha/2)\phi_{n}(u) \tag{14-6}$$

其中：

$$\phi_{n}(u)=\frac{1}{\sqrt{2^{n}\sqrt{\pi}n!}}H_{n}(u)\exp\left(-u^{2}/2\right) \tag{14-7}$$

$H_{n}(x)$ 是厄米多项式函数。$\phi_{n}(u)$ 被称为归一化的 n 阶厄米-高斯函数。系数 C_{n} 由下面的积分确定：

$$C_{n}=\int_{-\infty}^{+\infty} f(x)\phi_{n}(x)\mathrm{d}x \tag{14-8}$$

Yang 等人提出了一种基于卷积算法计算分数傅里叶变换的方案。非整数阶分数傅里叶变换积分表达式（14-3）可以写成下面的形式：

$$\mathrm{F}^{\alpha}\{f(x)\}(u)=A_{\alpha}\exp\left[-\mathrm{i}\pi\tan\left(\frac{\alpha\pi}{4}\right)u^{2}\right]\int_{-\infty}^{+\infty} f(x)\exp\left[-\mathrm{i}\pi\tan\left(\frac{\alpha\pi}{4}\right)x^{2}\right]\exp\left[\mathrm{i}\pi\csc\left(\frac{\alpha\pi}{2}\right)(u-x)^{2}\right]\mathrm{d}x \tag{14-9}$$

上式中的积分是一个卷积形式，即：

$$\int_{-\infty}^{+\infty} f(x)\exp\left[-\mathrm{i}\pi\tan\left(\frac{\alpha\pi}{4}\right)x^2\right]\exp\left[\mathrm{i}\pi\csc\left(\frac{\alpha\pi}{2}\right)(u-x)^2\right]\mathrm{d}x = f'(x)\otimes B_\alpha(x) , \quad (14\text{-}10)$$

这里符号“⊗”表示卷积运算，其中：

$$f'(x)=f(x)\exp\left[-\mathrm{i}\pi\tan\left(\frac{\alpha\pi}{2}\right)x^2\right], \quad B_\alpha(x)=\exp\left[\mathrm{i}\pi\csc\left(\frac{\alpha\pi}{2}\right)x^2\right] \quad (14\text{-}11)$$

令：

$$B'_\alpha(x)=\exp\left[-\mathrm{i}\pi\tan\left(\frac{\alpha\pi}{4}\right)x^2\right] \quad (14\text{-}12)$$

则转换为：

$$f'(x)=f(x)B'_\alpha(x) \quad (14\text{-}13)$$

通过上面的推导并结合卷积理论，公式（14-3）可以写为下面的形式：

$$\mathrm{F}^{\alpha}\{f(x)\}(u)=A_\alpha B'_\alpha(u)\left[f'_\alpha(x)\otimes B_\alpha(x)\right]=A_\alpha B'_\alpha(u)\mathrm{F}^{-1}\left\{\mathrm{F}\left[f'_\alpha(x)\right](u')\mathrm{F}\left[B_\alpha(x)\right](u')\right\}(u) \quad (14\text{-}14)$$

根据式（14-14），可以使用傅里叶变换计算分数傅里叶变换。在离散情况下，可以调用 MATLAB 的函数 fft 或者 fft2 来计算离散分数傅里叶变换。同时计算量因使用快速傅里叶变换而降低很多。特别地，在式（14-14）中 $\mathrm{F}\left[B_\alpha(x)\right](u')$ 可以得出解析的结果，即：

$$\mathrm{F}\left[B_\alpha(x)\right](u')=\exp\left[-\mathrm{i}\pi\sin\left(\frac{\alpha\pi}{2}\right)u'^2\right] \quad (14\text{-}15)$$

二维分数傅里叶变换的表达式为：

$$\mathrm{F}^{\alpha x,\,\alpha y}\{f(x,y)\}(u,v)=\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(x,y)K_{\alpha x,\,\alpha y}(x,y,u,v)\mathrm{d}x\mathrm{d}y \quad (14\text{-}16)$$

其中：

$$K_{\alpha x,\,\alpha y}=A_{\alpha x}A_{\alpha y}\exp\left[\mathrm{i}\pi\left(u^2\cot\theta_{\alpha x}+v^2\cot\theta_{\alpha y}-2xu\csc\theta_{\alpha x}-2yv\csc\theta_{\alpha y}+x^2\cot\theta_{\alpha x}+y^2\cot\theta_{\alpha y}\right)\right] \quad (14\text{-}17)$$

αx 和 αy 分别是水平和竖直方向的分数阶，$A_{\alpha x}$，$A_{\alpha y}$，$\theta_{\alpha x}$ 和 $\theta_{\alpha y}$ 定义类似于公式（14-5）。

根据上面介绍的计算过程，参见公式（14-5），给出离散形式下的分数傅里叶变换程序，下面是一维变换的 MATLAB 程序：

```
function ff=frftc1d(f,ax);
ax=ax*pi/2;                    % 转化为变换角θα
N = length(f);                  % 计算信号长度
f=f(:)';                       % 把信号转换为行向量
x = ([0:N-1]-[N-1]/2)/sqrt(N);  % 获得采样点
Ba = exp(i*pi*x.^2*csc(ax));    % 计算函数Bα(x)
Bpa = exp(-i*pi*x.^2*tan(ax/2));  % 计算函数B'α(x)
fa = f.*Bpa;                        % 计算函数f'(x)
C1 = fftshift(fft(fftshift(fa)));          % 计算傅里叶变换
C2 = exp(-i*pi*x.^2*sin(ax));         % 函数Bα(x)的傅里叶变换结果
CC = fftshift(ifft(fftshift(C1.*C2)));     % 求逆傅里叶变换
ff = Bpa.*CC;                          % 计算函数B'α(x)与卷积结果的乘积
ff = ff*sqrt(1-i*cot(ax));                 % 乘系数
```



ff 是输出的分数傅里叶变换结果。f 是输入的信号。ax 是变换的分数阶。

例 14-4：计算下面函数的分数傅里叶变换：

$y(t)=\mathrm{rect}(t)=\begin{cases}1, & |t|<1\\0, & |t|\geq 1\end{cases}$，其中 t 的取值范围是 $[-4,4]$，采样步长为 $\Delta t=1/16$，变换的分数阶是 $\alpha=0.7$。

分析：首先对函数进行离散采样并获得相应的函数值，然后调用前面编写的 frftc1d 函数计算离散分数傅里叶变换。其中离散采样可使用函数 linspace 来取得等间距离散的采样点。

首先生成函数 $y(t)$ 的离散形式，再调用前面给出的函数 frftc1d 进行分数傅里叶变换。相应计算和绘图程序如下：

```
t = linspace(-4,4,129);     % 计算采样点
yt = zeros(size(t));        % 生成与 t 相同大小的全 0 向量
yt(abs(t)<=1)=1;            % 得出输入函数的采样点
ff = frftc1d(yt,0.7);       % 进行分数傅里叶变换
figure;subplot(131);        %生成子图
plot(t,yt,'k');axis square;ylim([-0.2,1.2]);  % 画曲线，设置坐标轴为方形，设置 Y 轴范围
xlabel({'{\itt}','(a)'},'Fontsize',14,'Fontname','Times New Roman');  %标注 X 轴
title('signal','Fontsize',14);     % 标出图题
set(gca,'Fontsize',12);            % 设置坐标轴字体大小
subplot(132);plot(t,abs(ff),'k');axis square;   % 画曲线，设置坐标轴为方形
title('amplitude','Fontsize',14);  % 标出图题
xlabel({'{\itu}','(b)'},'Fontsize',14,'Fontname','Times New Roman');  %标注 X 轴
set(gca,'Fontsize',12);            % 设置坐标轴字体大小
subplot(133);plot(t,angle(ff),'kx');axis square; % 画曲线，设置坐标轴为方形
xlabel({'{\itu}','(c)'},'Fontsize',14,'Fontname','Times New Roman');  %标注 X 轴
title('phase','Fontsize',14);      % 标出图题
set(gca,'Fontsize',12);            % 设置坐标轴字体大小
```

执行上面的程序可以得到分数傅里叶变换结果，如图 14.3 所示。其中振幅部分在中心部分较高，两侧较低，而位相取值比较散乱。



用户可变换不同的分数阶来查看结果的差异，上述程序在光盘的\Ch14 文件夹下可以找到。

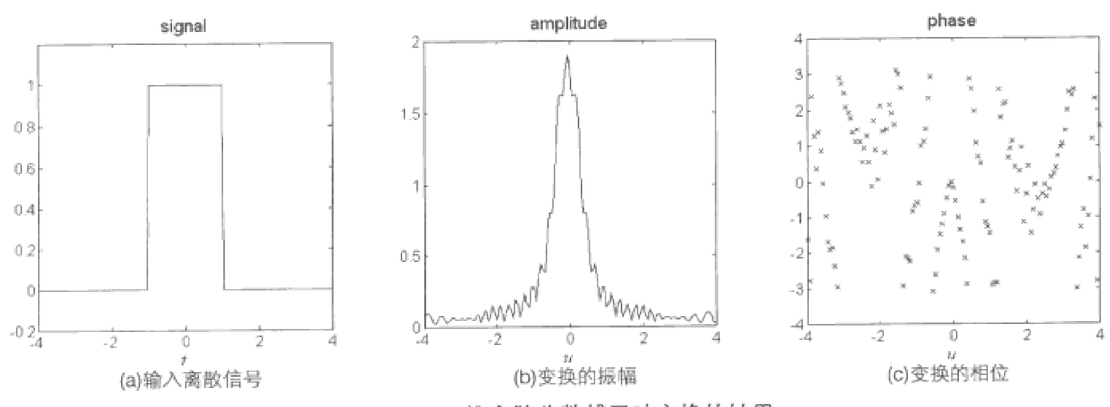


图 14.3 一维离散分数傅里叶变换的结果

二维变换的程序可以把一维变换实现程序中的函数 fft 变为 fft2、向量变为矩阵进行计算。详细

程序不再赘述，其保存在光盘的\Ch14 文件夹下，文件名为 frftc2d.m。该程序调用格式为：

```
ff = frftc2d(f,ax,ay);
```

参数说明：ff 是输出的变换结果。f 是一个矩阵，表示输入的二维函数的离散形式，也可以是数字图像对应的矩阵。ax 和 ay 分别是 x 方向和 y 方向的变换阶数，它们可以选择不同的数值。

例 14-5：对二维矩形函数进行分数傅里叶变换要求 x 和 y 在区间 $[-3,3]$ 上等间距采样且采样点数为 256，其中分数阶为 $\alpha x = 0.6$，$\alpha y = 0.4$。

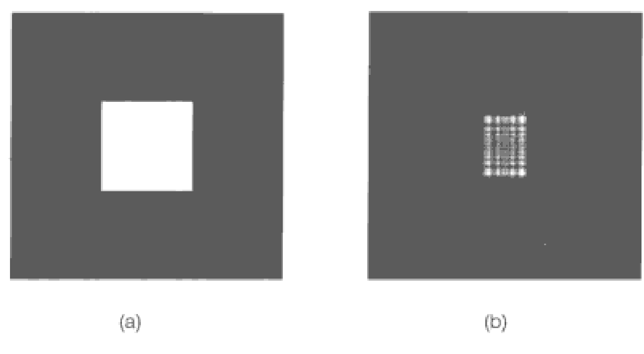
$$y(x,y)=\begin{cases}1, & |x|\le 1 \text{ and } |y|\le 1\\ 0, & \text{otherwise}\end{cases}$$

分析：类似于一维变换的计算，先生成离散的采样点和采样函数值，然后调用函数 frftc2d 计算二维离散分数傅里叶变换。

首先生成二元函数 $f(x,y)$ 的离散形式对应的矩阵，然后调用前面给出的函数 frftc2d 计算二维分数傅里叶变换。相应的 MATLAB 程序如下：

```
[x,y] = meshgrid(linspace(-3,3,256));   % 生成 X 轴和 Y 轴采样点数据
X = zeros(size(x));                   % 生成与 x 大小相同的全 0 矩阵
X(abs(x)<=1&abs(y)<=1)=1;            % 计算相应位置处的函数值
f1 = frftc2d(X,0.6,0.4);                % 计算二维离散分数傅里叶变换
figure;subplot(121);imshow(abs(X),[]);                % 绘图
xlabel('(a)','Fontsize',14,'Fontname','Times New Roman');  %标注 X 轴
subplot(122);imshow(abs(f1),[]);                     % 绘图
xlabel('(b)','Fontsize',14,'Fontname','Times New Roman');  %标注 X 轴
```

所得图形如图 14.4 所示，图(b)给出了变换的结果，表明主要能量集中在重要部分，其形状为一个矩形。



(a) (b)

图 14.4 二维离散分数傅里叶变换结果

除了上面提到的利用卷积算法计算分数傅里叶变换的程序外，还可以利用分数傅里叶变换的本征函数——厄米-高斯函数来计算，依据是公式（14-6）。利用厄米函数的递推关系可以得到变换的核矩阵，然后计算核矩阵与输入向量的积就可以得到分数傅里叶变换了。实现程序读者可以参阅光盘中的 frft_PXJ.m 文件。

在网络上还可以下载到几个一维离散分数傅里叶变换的 MATLAB 计算程序，如网站 http://www.cs.kuleuven.be/~nalag/research/software/FRFT/。

这些程序的用法这里不再赘述了，读者可从相应的文献和程序注释信息了解这个程序。

14.2 菲涅尔变换

菲涅尔（Fresnel）变换定义为：

$$G(u)=F_A\left[g(x)\right](u)=\int_{-\infty}^{+\infty}g(x)\exp\left[\mathrm{i}\pi\frac{(x-u)^2}{A}\right]\mathrm{d}x \tag{14-18}$$

其中 $g(x)$ 和 $G(u)$ 分别是菲涅尔变换的输入和输出。F_A 表示菲涅尔变换，A 是菲涅耳变换的参数。公式（14-18）可以表示为一个卷积的形式：

$$G(u)=g(x)\otimes\exp\left(\mathrm{i}\pi x^2/A\right) \tag{14-19}$$

符号“⊗”表示卷积。卷积可以利用傅里叶变换来计算，即：

$$G(u)=\mathrm{F}^{-1}\left\{\mathrm{F}\left[g(x)\right](u')\mathrm{F}\left[\exp\left(\mathrm{i}\pi x^2/A\right)\right](u')\right\}(u) \tag{14-20}$$

其中 $\mathrm{F}\left[\exp\left(\mathrm{i}\pi x^2/A\right)\right](u')$ 可以得到如下解析结果：

$$\mathrm{F}\left[\exp\left(\mathrm{i}\pi x^2/A\right)\right](u')=\exp\left(-\mathrm{i}\pi A u'^2\right) \tag{14-21}$$

根据公式（14-20）和公式（14-21）可以写出菲涅尔变换离散形式的计算程序，其中正/逆傅里叶变换可以调用函数 fft 和 ifft。一维离散菲涅尔变换的程序如下：

```
function ff=fresnel_1d(f,A);
warning off;
N = length(f);                          % 计算信号长度
f=f(:)';                               % 把信号转换为行向量
x = ([0:N-1]-[N-1]/2)/sqrt(N);          % 获得采样点
Ex = exp(-i*pi*x.^2*A);            % 计算函数 exp(i*pi*x^2/A)的傅里叶变换
C1 = fftshift(fft(fftshift(f)));         % 计算傅里叶变换
ff = fftshift(ifft(fftshift(C1.*Ex)));  % 求逆傅里叶变换
```

参数说明： ff 为输出的菲涅尔变换结果。f 是输入数据。A 是变换的参数。

例 14-6： 计算下面函数的菲涅尔变换，其中采样步长为 $\Delta x=0.03$，菲涅尔变换的参数为 $A=3$。

$$y(x)=\begin{cases}1, & |x|\le 1\\ 0, & 1<|x|\le 3\end{cases}$$

分析： 在进行菲涅尔变换之前，先对函数离散采样得到采样点位置处的函数值，然后调用前面定义的离散菲涅尔函数进行计算，即可得到该函数的变换结果。

首先给出输入函数 $y(x)$ 的离散形式，即向量，然后调用函数 fresnel_1d 计算菲涅尔变换。根据这个过程可以写出相关的计算和绘图程序，内容如下：

```
A=3;                       % 对参数 A 进行赋值
x=linspace(-3,3,201);        % 离散采样
yx=zeros(1,201);           % 生成全 0 向量
yx(abs(x)<=1)=1;           % 得到采样点处的函数值
fy=fresnel_1d(yx,A);        % 进行菲涅尔变换
figure; subplot(131);
plot(x,yx,'k');ylim([-0.2,1.2]);xlim([-3,3]);axis square;  % 画出输入函数的曲线
xlabel('(a)');
subplot(132);
plot(x,real(fy),'k');xlim([-3,3]);axis square;              % 画出变换结果的实部
xlabel('(b)');
```

```
subplot(133);
plot(x,imag(fy),'k');xlim([-3,3]);axis square;          % 画出变换结果的虚部
xlabel('(c)');
```

上述程序执行后会得到如图 14.5 所示的曲线，从图(b)和图(c)中可以看出较大的数值分布在实部和虚部的中央处。

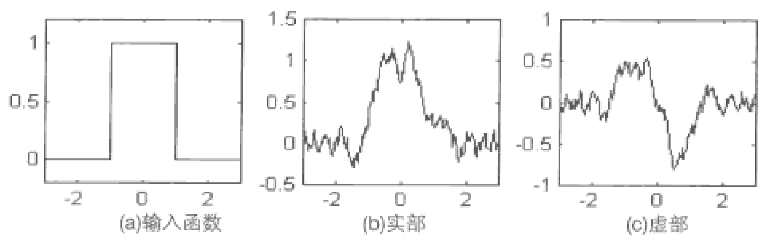


图 14.5　一维离散菲涅尔变换的结果

二维菲涅尔变换的表达式为：

$$G(u,v)=\mathrm{F}^{-1}\left\{\mathrm{F}\left[g(x,y)\right](u',v')\exp\left[\mathrm{i}\pi A\left(u'^2+v'^2\right)\right]\right\}(u,v) \tag{14-22}$$

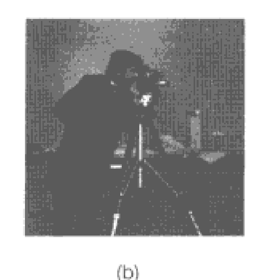
二维离散菲涅尔变换的程序可以根据式（14-22）编写，具体如下：

```
function ff=fresnel_2d(f,A);
[M,N] = size(f);                    % 计算信号长度
[x,y] = meshgrid(([0:N-1]-[N-1]/2)/sqrt(N),([0:M-1]-[M-1]/2)/sqrt(M));   % 获得采样点
Exy = exp(-i*pi*[x.^2+y.^2]*A);    % 计算函数 exp(i*pi*[x^2+y^2]/A)的傅里叶变换
C1 = fftshift(fft2(fftshift(f)));           % 计算傅里叶变换
ff = fftshift(ifft2(fftshift(C1.*Exy)));  % 求逆傅里叶变换
```

例 14–7：计算下面两幅图像（如图 14.6 所示）对应的二维离散菲涅尔变换，其中参数取值为 $A=4$。



(a)



(b)

图 14.6　输入图像

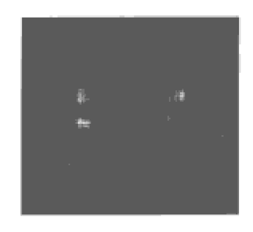


图 14.6 中图(a)是在光盘中\Ch14 文件夹下的 EF.bmp 文件，图(b)是自带的图像。

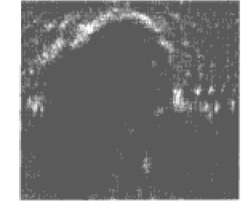
分析：利用函数 imread 可以读入这两幅图像而到相应的像素矩阵，其中用户需要把 EF.bmp 文件复制到 MATLAB 当前路径下，然后调用前面定义的函数 fresnel_2d 就可以进行菲涅尔变换的计算了。计算程序如下：

```
A=4;                              % 对变换的参数赋值
f1=double(imread('EF.bmp'));          % 读入第一个图像
f2=double(imread('cameraman.tif'));   % 读入第二个图像
F1=fresnel_2d(f1,A);              % 计算离散菲涅尔变换
F2=fresnel_2d(f2,A);              % 计算离散菲涅尔变换
subplot(121);imshow(abs(F1),[]);      % 绘图
xlabel('(a)','Fontsize',16); % 标注
subplot(122);imshow(abs(F2),[]);      % 绘图
xlabel('(b)','Fontsize',16);          % 标注
```

所得图形如图 14.7 所示，从变换结果可以看出该变换使输入图像变得模糊了，原输入图像很难分辨。



(a) "EF" 字母对应的结果



(b)摄影师的变换结果

图 14.7 菲涅尔变换的结果

14.3 Hartley 变换

一维和二维 Hartley 变换的数学定义如下：

$$Q_1(u)=\mathrm{H}\left[q_1(x)\right](u)=\int_{-\infty}^{+\infty}q_1(x)\mathrm{cas}(2\pi ux)\mathrm{d}x \tag{14-23}$$

$$Q_2(u,v)=\mathrm{H}\left[q_2(x,y)\right](u,v)=\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}q_2(x,y)\mathrm{cas}\left[2\pi(ux+vy)\right]\mathrm{d}x\mathrm{d}y \tag{14-24}$$

其中 $Q_1(u)$ 和 $Q_2(u,v)$ 分别是一维和二维 Hartley 变换的输出结果。$q_1(x)$ 和 $q_2(x,y)$ 分别是一维和二维输入函数。符号"H"代表 Hartley 变换。cas 的含义是 $\mathrm{cas}x=\cos x+\sin x$。

从上面的定义可以看出 Hartley 变换实质上就是傅里叶变换的实部和虚部相加。另外，傅里叶变换的本征值为 $\{-1,1,-\mathrm{i},\mathrm{i}\}$，可以推知 Hartley 变换的本征值为 $\{-1,1\}$。因为一个函数经过 4 次连续的傅里叶变换得到函数自身，类似地可以知道连续两次 Hartley 变换即可得到函数自身。进一步可以得到 Hartley 变换的逆变换就是其自身。

根据上面的分析可以得到离散 Hartley 变换的计算程序，一维变换的程序具体内容如下：

```
function Q=hartley1(q);
N=length(q);                     % 计算向量中元素数目
Q=fft(q);                        % 离散傅里叶变换
Q=[real(Q)+imag(Q)]/sqrt(N);     % 取实部与虚部之和
```

参数说明： Q 是输出的变换结果，正确的 Hartley 变换的频谱需要通过函数 fftshift 进行频移。q 表示输入的离散函数。

例 14-8： 计算下面函数的离散 Hartley 变换结果，其中采样步长为 $\Delta x=0.06$。

$$y(x)=\begin{cases}x, & |x\le 1| \\ 0, & 1<|x|\le 3\end{cases}$$

分析：首先对函数的定义区间离散取样，然后令区间内各位置函数值等于 0，对于区间$[-1,1]$内的函数值等于 1，再将函数与变量 x 相乘即可得到问题中的函数定义。接下来调用 hartley1 进行变换的计算。根据分析过程可以写出相应的计算程序，其内容如下：

```
x=linspace(-3,3,101);      % 计算采样点
y=zeros(size(x));          % 生成全 0 向量
y(abs(x)<=1)=1;            % 对规定范围内的数值设置为 1
y=y.*x;                    % 计算出函数在采样点的数值
hy=hartley1(y);            % 计算 Hartley 变换
hhy=hartley1(hy);          % 计算 Hartley 变换结果的 Hartley 变换，即逆变换
figure;s1=subplot(131);
plot(x,y,'k');axis square;xlim([min(x),max(x)]);    % 画出输入离散函数对应的曲线
xlabel('(a)','Fontsize',16);  % X 轴标注
s2=subplot(132);
plot(x,fftshift(hy),'k');axis square;xlim([min(x),max(x)]);  % 变换结果
xlabel('(b)','Fontsize',16);  % X 轴标注
s3=subplot(133);
plot(x,hhy,'k');axis square;xlim([min(x),max(x)]);   % 逆变换
xlabel('(c)','Fontsize',16);
set([s1,s2,s3],'Fontsize',12);
```

所得结果如图 14.8 所示。

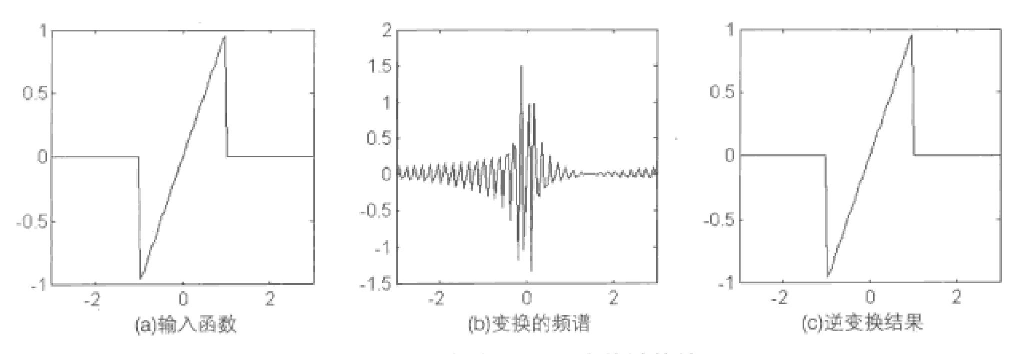


图 14.8 一维离散 Hartley 变换计算结果

从图 14.8 可以看出连续两次 Hartley 变换即可得到原来的输入函数，验证了前面提到的结论。

根据式（14-24），可以写出二维 Hartley 变换的程序，具体如下：

```
function Q=hartley2(q);
[M,N]=size(q);        % 计算向量中元素的行数和列数
Q=fft2(q);            % 离散傅里叶变换
Q=[real(Q)+imag(Q)]/sqrt(N*M);  % 取实部与虚部之和
```

参数说明：Q 和 q 分别是二维离散 Hartley 变换的输出和输入数据。对于二维 Hartley 变换同样需要利用函数 fftshift 来调整频谱的顺序。

例 14-9：计算如图 14.9 所示的图像的 Hartley 变换与逆变换。





图 14.9　字符“H”的图像

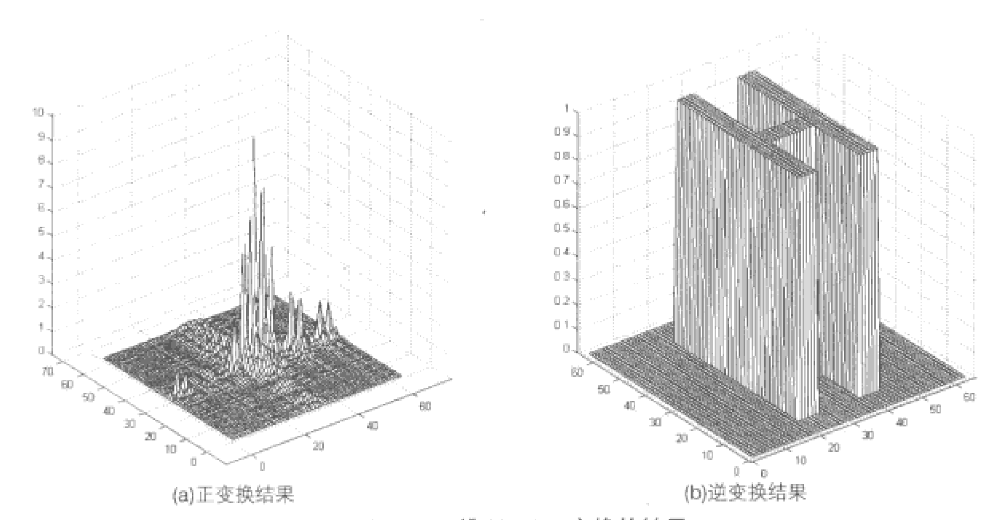


该文件保存于光盘中\Ch14 文件夹下的 hw.bmp 文件中。

分析：使用函数 imread 可以读入图像所对应的矩阵，然后调用上面的 hartley2 函数进行二维 Hartley 变换。这里利用 mesh 函数来显示频谱的输出图形。相应的实现程序如下：

```
q1=double(imread('hw.bmp'));  % 读入图像
Q1=hartley2(q1);              % 进行二维 Hartley 变换
QQ1=hartley2(Q1);             % 再进行一次 Hartley 变换
figure;subplot(121);
mesh(abs(fftshift(Q1))); axis vis3d;   % 绘图
xlabel('(a)','Fontsize',18);       % X 轴标注
subplot(122);
mesh(QQ1);axis vis3d;         % 绘图
set(gcf,'Color','w');            % 设置背景为白色
xlabel('(b)','Fontsize',18);       % X 轴标注
```

所得图形如图 14.10 所示。



(a)正变换结果　　(b)逆变换结果

图 14.10　二维 Hartley 变换的结果

从图 14.10 可以看出，Hartley 变换在中心处的频谱分量较大，连续两次变换就得到原输入图像。

14.4　离散正/余弦变换

目前离散正余弦变换的版本有如下 4 种形式：

$$C_N^1(m,n)=\sqrt{\frac{2}{N-1}}\left[k_m k_n \cos\left(\frac{mn\pi}{N-1}\right)\right],\quad m,n=0,1,\ldots,N-1 \tag{14-25}$$

$$C_N^2(m,n)=\sqrt{\frac{2}{N}}\left[k_m \cos\left(m\pi\frac{n+1/2}{N}\right)\right],\quad m,n=0,1,\ldots,N-1 \tag{14-26}$$

$$C_N^3(m,n)=\sqrt{\frac{2}{N}}\left[k_n \cos\left(n\pi\frac{m+1/2}{N}\right)\right],\quad m,n=0,1,\ldots,N-1 \tag{14-27}$$

$$C_N^4(m,n)=\sqrt{\frac{2}{N}}\left\{\cos\left[\pi\frac{(m+1/2)(n+1/2)}{N}\right]\right\},\quad m,n=0,1,\ldots,N-1 \tag{14-28}$$

其中C_N^s($s=1,2,3,4$)表示离散余弦变换的核矩阵，其中的s表示不同类型的离散余弦变换，N对应于信号的长度。对应的离散正弦变换也有 4 种形式，即：

$$S_N^1(m,n)=\sqrt{\frac{2}{N+1}}\left[\sin\left(\frac{mn\pi}{N+1}\right)\right],\quad m,n=1,\ldots,N \tag{14-29}$$

$$S_N^2(m,n)=\sqrt{\frac{2}{N}}\left[k_m \sin\left(m\pi\frac{n-1/2}{N}\right)\right],\quad m,n=1,\ldots,N \tag{14-30}$$

$$S_N^3(m,n)=\sqrt{\frac{2}{N}}\left[k_n \sin\left(n\pi\frac{m-1/2}{N}\right)\right],\quad m,n=1,\ldots,N \tag{14-31}$$

$$S_N^4(m,n)=\sqrt{\frac{2}{N}}\left\{\sin\left[\pi\frac{(m-1/2)(n-1/2)}{N}\right]\right\},\quad m,n=1,\ldots,N \tag{14-32}$$

上面 8 个公式中，$k_0=k_N=1/\sqrt{2}$，其他的k_m和k_n等于 1。根据上面的定义公式可以写出对应的变换程序，其中以C_N^1为例，根据公式（14-25）可以写出离散余弦变换的程序，其内容如下：

```
function y=dct1(x);              % 1 型离散余弦变换
N=length(x);                   % 计算参数 N
x=x(:);                        % 把信号转为列向量
[n,m]=meshgrid(0:N-1);          % 生成脚标矩阵
km=ones(N);                    % 生成系数 km
kn=km;                         % 生成系数 kn
km(1,:)=1/sqrt(2);             % 设置系数 km 的取值
km(end,:)=1/sqrt(2);           % 设置系数 km 的取值
kn(:,1)=1/sqrt(2);             % 设置系数 kn 的取值
kn(:,end)=1/sqrt(2);           % 设置系数 kn 的取值
C1N=sqrt(2/[N-1])*ones(N);  % 初始化变换核矩阵，所有元素都等于 sqrt(2/[N-1])
C1N=C1N.*kn.*km.*cos(m.*n*pi/[N-1]);  % 计算核矩阵
y=C1N*x;                             % 进行离散余弦变换
```

参数说明：上述程序中 y 是变换的输出结果。x 是输入的离散函数。其他几个函数可以类似地写出，相应程序保存在光盘\Ch14 文件夹中，文件名依次为 dct1.m，dct2.m，dct3.m，dct4.m，dst1.m，dst2.m，dst3.m 和 dst4.m。相应程序不在这里描述，参数意义都相同。

例 14–10：计算例 14-8 中函数相应的离散余弦变换和离散正弦变换。

分析： 首先得到输入函数的离散形式，然后依次调用上面的 dct1.m，dct2.m，dct3.m，dct4.m，dst1.m，dst2.m，dst3.m 和 dst4.m 程序就可以计算离散正/余弦变换的结果。

根据分析可以写出相应计算变换的程序，具体内容如下：

```
t=linspace(-3,3,101);      % 自变量离散采样
x=zeros(size(t));          % 计算输入函数的离散形式
x(abs(t)<=1)=1;            % 限定范围内的数值等于 1
x=x.*t;                    % 计算输入函数的离散形式
yc1=dct1(x);               % 1 型离散余弦变换
yc2=dct2(x);               % 2 型离散余弦变换
yc3=dct3(x);               % 3 型离散余弦变换
yc4=dct4(x);               % 4 型离散余弦变换
ys1=dst1(x);               % 1 型离散正弦变换
ys2=dst2(x);             % 2 型离散正弦变换
ys3=dst3(x);             % 3 型离散正弦变换
ys4=dst4(x);             % 4 型离散正弦变换
subplot(241);plot(t,yc1,'k');  % 绘图
text(1,1.5,'{\itC}_{\itN}^1','Fontname','Times new roman','Fontsize',14);  % 标注
subplot(242);plot(t,yc2,'k');  % 绘图
text(1,1.5,'{\itC}_{\itN}^2','Fontname','Times new roman','Fontsize',14);  % 标注
subplot(243);plot(t,yc3,'k');  % 绘图
text(1,1.5,'{\itC}_{\itN}^3','Fontname','Times new roman','Fontsize',14);  % 标注
subplot(244);plot(t,yc4,'k');  % 绘图
text(1,1.5,'{\itC}_{\itN}^4','Fontname','Times new roman','Fontsize',14);  % 标注
subplot(245);plot(t,ys1,'k');  % 绘图
text(1,1.5,'{\itS}_{\itN}^1','Fontname','Times new roman','Fontsize',14);  % 标注
subplot(246);plot(t,ys2,'k');  % 绘图
text(1,2.3,'{\itS}_{\itN}^2','Fontname','Times new roman','Fontsize',14);  % 标注
subplot(247);plot(t,ys3,'k');  % 绘图
text(1,1.5,'{\itS}_{\itN}^3','Fontname','Times new roman','Fontsize',14);  % 标注
subplot(248);plot(t,ys4,'k');  % 绘图
text(1,1.5,'{\itS}_{\itN}^4','Fontname','Times new roman','Fontsize',14);  % 标注
```

输出图形如图 14.11 所示。

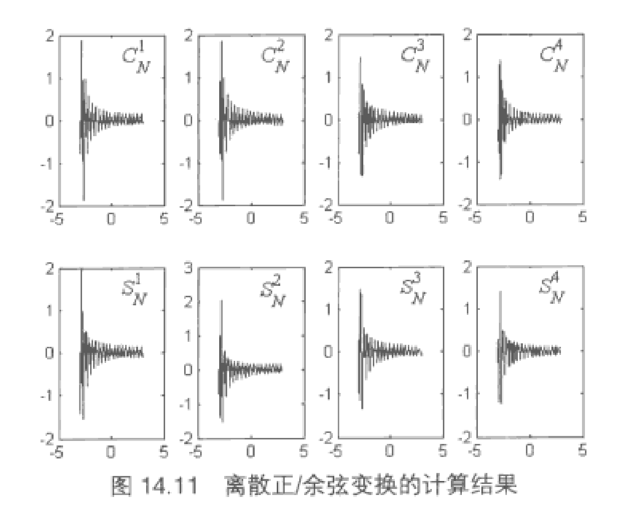


图 14.11　离散正/余弦变换的计算结果

从图 14.11 可以看出不同版本的离散正/余弦变换结果大体相似，但在细节上存在差异。

对于二维变换的程序，可以分别生成行向量和列向量对应的变换核矩阵，然后利用核矩阵和输

入矩阵相乘得到二维变换结果，这里以 1 型离散余弦变换为例给出相应实现程序，二维程序的基本思路是利用两个变换核矩阵分别左乘和右乘矩阵两侧，具体程序如下：

```
function y=dct1_2d(x);      % 1 型二维离散余弦变换
[M,N]=size(x);              % 计算参数 N
C1M=kernelt(M);             % 得到各列变换对应的核矩阵
C1N=kernelt(N);             % 得到各行变换对应的核矩阵
y=C1M*x*C1N;                % 进行离散余弦变换
function C1N=kernelt(N);    % 计算核矩阵的子程序
[n,m]=meshgrid(0:N-1);      % 生成脚标矩阵
km=ones(N);                 % 生成系数 km
kn=km;                      % 生成系数 kn
km(1,:)=1/sqrt(2);          % 对系数向量 km 赋值
km(end,:)=1/sqrt(2);        % 对系数向量 km 赋值
kn(:,1)=1/sqrt(2);          % 对系数向量 kn 赋值
kn(:,end)=1/sqrt(2);        % 对系数向量 kn 赋值
C1N=sqrt(2/[N-1])*ones(N);  % 初始化变换核矩阵，所有元素都等于 sqrt(2/[N-1])
C1N=C1N.*kn.*km.*cos(m.*n*pi/[N-1]);  % 计算核矩阵
```

参数说明： y 和 x 分别是变换的输出和输入矩阵。其中核矩阵通过一个子函数计算，即 kernelt。其他程序保存于光盘中\Ch14 文件夹下，文件名依次为 dct2_2d.m，dct3_2d.m，dct4_2d.m，dst1_2d.m，dst2_2d.m，dst3_2d.m 和 dst4_2d.m。

例 14-11： 计算下面图案的离散余弦变换，如图 14.12 所示。

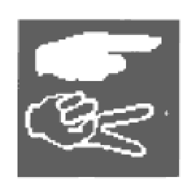


图 14.12 进行离散余弦变换的图案



该图案对应的图片文件保存在光盘中的\Ch14 文件夹下，文件名为 fp.bmp。

分析： 先利用 imread 函数把图像对应的数据读入 MATLAB，然后调用上面定义的离散余弦变换函数 dct1_2d 来计算即可。实现程序如下：

```
A=double(imread('fp.bmp'));  % 读入图像数据并转化为 double 型数据
C1=dct1_2d(A);               % 1 型二维离散余弦变换
mesh(C1);                    % 绘图
colormap([0,0,0])
```

运行上述程序结果如图 14.13 所示。

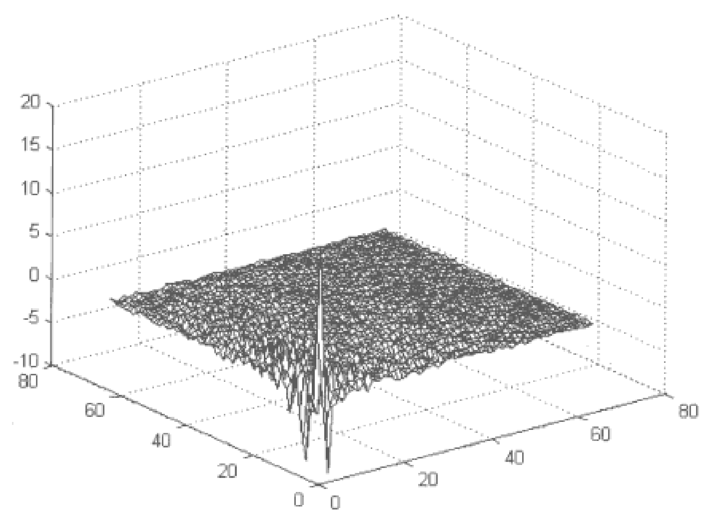


图 14.13　二维离散余弦变换结果

从图 14.13 可以看出，变换结果中在$(0,0)$附近的数值比较大，其他位置的数值接近于 0。其他类型的余弦变换和离散正弦变换的二维结果，读者可以自行计算一下。

14.5　分数随机变换

本节讨论分数阶随机变换的定义和程序实现。离散随机变换的数学定义步骤如下：

step 1　产生随机的对称实数矩阵Q（这里要求矩阵Q是正定的）。

step 2　计算矩阵Q的本征向量矩阵V，它们将作为随机变换的本征向量。

step 3　按下述方式得到随机变换的本征值对角方矩阵：

$$D^{\alpha}=\mathrm{diag}\left(-2\mathrm{i}\pi\alpha[0,1,\ldots,N-1]\right) \tag{14-33}$$

其中函数 diag 表示由向量生成对角矩阵，α是变换的分数阶。

step 4　得到变换的核矩阵R^{α}：

$$\mathrm{R}^{\alpha}=VD^{\alpha}V^{t} \tag{14-34}$$

其中符号“t”表示矩阵的转置运算。通过核矩阵R^{α}和输入信号或者图像相乘就可以进行离散分数随机变换。

离散分数随机变换具有可加性、能量守恒、线性等性质，关于该变换的应用，可以参阅相关的文献。根据上面的描述可以得到一维离散分数随机变换的程序：

```
function y=dfrt(x,alpha,s);  % 离散分数随机变换
N=length(x);                % 获得元素数目
rand('state',s);                % 固定随机数的种子
P=rand(N);                 % 生成一个随机矩阵
Q=(P+P')/2;                % 得到实对称矩阵
if rank(Q)<N;      % 验证矩阵 Q 是否正定
   error('???   Matrix "Q" is not full rank matrix');
   output=[];
end
```

```
[V,D]=eig(Q);     % 计算一组本征向量
D=diag(exp(-2*i*pi*([0:N-1])*alpha));  % 本征值矩阵
T=V*D*V';         % 变换的核矩阵
x=x(:);           % 将x转化为列向量
y=T*x;            % 得到变换结果
```

参数说明：y 是变换的输出结果。alpha 是变换的分数阶。s 是控制随机数状态的参数。当变换的程序提示"Matrix "Q" is not full rank matrix"时，变换将不能进行，此时用户需要改变参数 s 的数值来得到一个正定的矩阵 Q。

例 14-12：计算下面函数的离散分数随机变换：

$x(t)=\begin{cases}1, & |t|\le 1\\ 0, & 1<|t|\le 2\end{cases}$，其中采样步长为 $\Delta t=0.05$，分数阶选择为 $\alpha=0.2,0.5$。

分析：首先对函数 $x(t)$ 进行离散采样并计算相应的函数值，然后调用前面定义的函数 dfrt 来计算离散分数随机变换。显示结果时选择实部和虚部分别显示的形式。而随机变换的参数 s 取值为 1021。

其求解程序如下：

```
t=linspace(-2,2,81);            % 自变量离散采样
x=zeros(size(t));
x(abs(t)<=1)=1;                 % 计算出对应位置的函数值
y1=dfrt(x,0.2,1021);            % 计算分数阶等于 0.2 时的输出结果
y2=dfrt(x,0.5,1021);            % 计算分数阶等于 0.5 时的输出结果
s1=subplot(141);plot(t,real(y1),'k');axis square;  % 绘图，变换的实部
xlabel('(a)','Fontsize',12);    % 标注
s2=subplot(142);plot(t,imag(y1),'k');axis square;  % 绘图，变换的虚部
xlabel('(b)','Fontsize',12);    % 标注
s3=subplot(143);plot(t,real(y2),'k');axis square;   % 绘图，变换的实部
xlabel('(c)','Fontsize',12);    % 标注
s4=subplot(144);plot(t,imag(y2),'k');axis square;  % 绘图，变换的虚部
xlabel('(d)','Fontsize',12);    % 标注
```

上述程序所得结果如图 14.14 所示。从图中可以看出，变换输出结果的实部和虚部都是随机的数据。图(d)中的数据都非常小。这是因为在式（14-31）中，当 α 等于 0.5 时，矩阵 D^{α} 是实数，从而矩阵 R^{α} 是实数。这里输入信号 $x(t)$ 是实数，所以输出结果也是实数。理论上图(d)中的数据应该等于 0，它们是很小的数，其来自于浮点型数据计算误差。

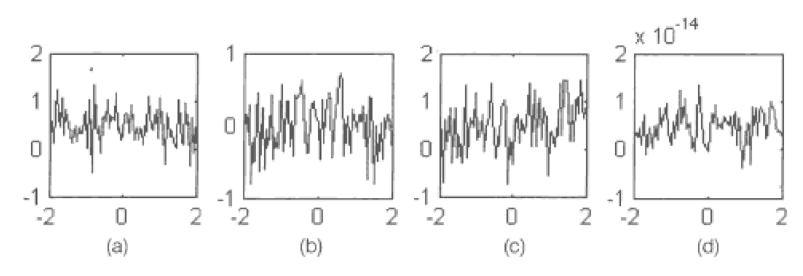


图 14.14 一维离散分数随机变换的结果：(a) 分数阶为 0.2 时变换结果的实部；(b) 分数阶为 0.2 时变换结果的虚部；(c) 分数阶为 0.5 时变换结果的实部；(d)分数阶为 0.5 时变换结果的虚部

二维变换的程序可以在一维的基础上通过对列向量和行向量分别变换得到，即用两个核矩阵分别乘到输入矩阵的两侧。根据这个思路可以写出相应的程序，内容如下：

```
function y=dfrt2(x,alpha,s);   % 二维离散分数随机变换
[M,N]=length(x);               % 获得矩阵的行数和列数
Tm=kernelT(M,alpha,s);         % 计算变换核矩阵
Tn=kernelT(N,alpha,s);          % 计算变换核矩阵
y=Tm*x*Tn;                     % 得到变换结果
function T=kernelT(N,alpha,s);  % 计算变换核矩阵的子程序
rand('state',s);                  % 固定随机数的种子
P=rand(N);                     % 生成一个随机矩阵
Q=(P+P')/2;                    % 得到实对称矩阵
if rank(Q)<N;                   % 验证矩阵 Q 是否正定
    error('???   Matrix "Q" is not full rank matrix');
    output=[];
end
[V,d]=eig(Q);                  % 计算出一组本征向量
D=diag(exp(-2*i*pi*([0:N-1])*alpha));  % 本征值矩阵
T=V*D*V';                     % 变换的核矩阵
```

参数说明： y 是变换的输出结果。x 是输入的矩阵。alpha 是变换的参数。参数 s 用来控制生成随机数的状态。同样对于二维变换，当分数阶等于 0.5 时可以得到实数的输出结果。上面的程序利用 kernelT 子函数来计算变换的核函数，可以用这个子函数计算列变换核矩阵和行变换核矩阵使用不同的分数阶。用户在上述程序中稍改动即可。光盘中\Ch14 文件夹下的 dfrt2xy.m 文件是实现水平和竖直方向不等分数阶变换的程序，程序内容不在这里显示。其调用格式为：

```
y=dfrt2xy(x,ax,ay,s);
```

其中 y 和 x 分别是变换的输出和输入矩阵，ax 和 ay 是变换在水平和竖直方向的分数阶，s 是控制生成随机数状态的参数。

例 14–13： 计算如图 14.15 所示图像对应矩阵的离散分数随机变换的结果。



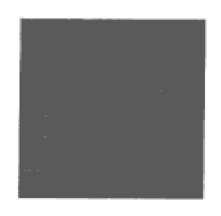
图 14.15　二值符号图案

分析： 首先利用函数 imread 读入图像对应的数据矩阵，然后调用前面定义的函数 dfrt2 计算离散分数随机变换的结果。根据分析可以写出对图像进行变换的程序，如下：

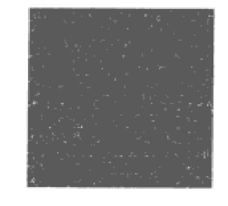
```
A=double(imread('ASC.bmp'));   % 读入图像数据
F1=dfrt2(A,0.3,150001);          % 进行分数阶为 0.3 的变换
M1=abs(F1);                    % 取振幅
A1=angle(F1);                   % 提取相位
A1(A1<0)=A1(A1<0)+pi*2;         % 把相位限制在[0, pi*2]
F2=dfrt2(A,0.7,150001);            % 进行分数阶为 0.7 的变换
M2=abs(F2);                    % 取振幅
A2=angle(F2);                   % 提取相位
A2(A2<0)=A2(A2<0)+pi*2;         % 把相位限制在[0, pi*2]
subplot(221);imshow(M1,[]);      % 绘制振幅 M1
```

```
xlabel('(a)','Fontsize',14,'Fontname','Times new roman'); % 标注
subplot(222);imshow(A1,[]);      % 绘制相位 A1
xlabel('(b)','Fontsize',14,'Fontname','Times new roman'); % 标注
subplot(223);imshow(M2,[]);      % 绘制振幅 M2
xlabel('(c)','Fontsize',14,'Fontname','Times new roman'); % 标注
subplot(224);imshow(A2,[]);      % 绘制相位 A2
xlabel('(d)','Fontsize',14,'Fontname','Times new roman'); % 标注
```

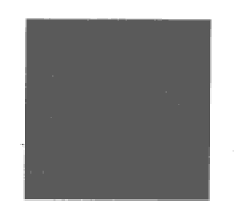
输出的图形如图 14.16 所示。



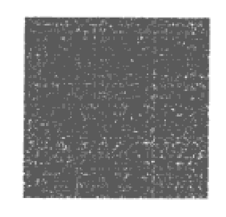
(a)分数阶为 0.3 时变换结果的振幅



(b)分数阶为 0.3 时变换结果的相位



(c)分数阶为 0.7 时变换结果的振幅



(d)分数阶为 0.7 时变换结果的相位

图 14.16 二维离散分数随机变换的结果

图 14.16 中图(a)和图(c)是一样的，这个可以根据公式（14-33）来证明。当两个分数阶 α 和 β 满足 $\alpha+\beta=1$ 关系时，变换 R^{α} 和 R^{β} 之间满足振幅相等、相位共轭的规律。对于水平和竖直不等的情况，可以使用前面提到的函数 dfrt2xy 来计算。

14.6 汉克尔（Hankel）变换

二维傅里叶变换在圆对称的情况下表达式可以写为：

$$G(\rho)=2\pi\int_0^{\infty} g(r)J_0(2\pi r\rho)r\mathrm{d}r \tag{14-35}$$

其中 $g(r)$ 为输入函数，$G(\rho)$ 是变换的输出函数。J_0 是零阶贝塞尔函数。r 和 ρ 分别是变换前后的空间坐标和频率坐标。表达式（14-35）定义的变换就是汉克尔（Hankel）变换（或者叫做零阶汉克尔变换），也称做傅里叶-贝塞尔变换（Fourier-Bessel transform）。当表达式（14-35）中的贝塞尔函数被 v 阶贝塞尔函数代替后，即：

$$G(\rho)=2\pi\int_0^{\infty} g(r)J_v(2\pi r\rho)r\mathrm{d}r \tag{14-36}$$

公式（14-36）被称为 v 阶贝塞尔函数，其中 v 是正整数。除了把 r 和 ρ（G 和 g）相互交换外，汉克尔变换的逆变换与公式（14-35）和（14-36）相同。对于实际问题的数值计算，公式（14-35）和（14-36）的积分不能积分到无穷，只能在有限大小的区间上进行，即 $[0,b]$。同样，频谱函数 $G(\rho)$

的定义区间也需要限定为$[0,\beta]$，b和β是正实数，根据具体问题确定，其中$b\beta$被称为空间带宽乘积（Space Bandwidth Produce）。下面来考虑汉克尔变换的数值计算。

例 14–14：计算下面函数的零阶汉克尔变换。

$$g_1(r)=\begin{cases}r, & 0\le r\le 1\\ 2-r, & 1<r<2\\ 0, & r\ge 2\end{cases}$$

分析：该问题中函数$g_1(r)$是一个分段函数，同时非零函数值限制在有限区间$[0,2]$内。公式（14-35）可以等价地写为下面的表达式：

$$G(\rho)=2\pi\left[\int_0^1 J_0(2\pi r\rho)r^2\mathrm{d}r+\int_1^2 J_0(2\pi r\rho)r(2-r)\mathrm{d}r\right] \tag{14-37}$$

对公式（14-37）进行积分计算获得函数值。此处用函数 quadgk 来计算公式（14-37）中的两个积分。带有确定解析表达式且非零值区间有限的汉克尔变换可以使用该思路计算，程序如下：

```
rho=linspace(0,4,200);          % 对频域坐标进行离散采样
for k=1:length(rho);            % 逐点计算频谱各位置处的函数值
    a=2*pi*rho(k);          % 计算系数
    G1(k)=quadgk(@(r)[r.^2.*besselj(0,r*a)],0,1);             % 计算第一个积分
    G1(k)=G1(k)+quadgk(@(r)[r.*(2-r).*besselj(0,r*a)],1,2);   % 计算第二个积分并累
加
end
plot(rho,G1*pi*2,'k');set(gca,'Fontsize',12);                  % 绘图
xlabel('\it\rho','Fontsize',14,'Fontname','Times new roman');     % 标注 X 轴
ylabel('{\itG}({\it\rho})','Fontsize',14,'Fontname','Times new roman');  % 标注 Y
轴
```

上述程序计算输出结果如图 14.17 所示。

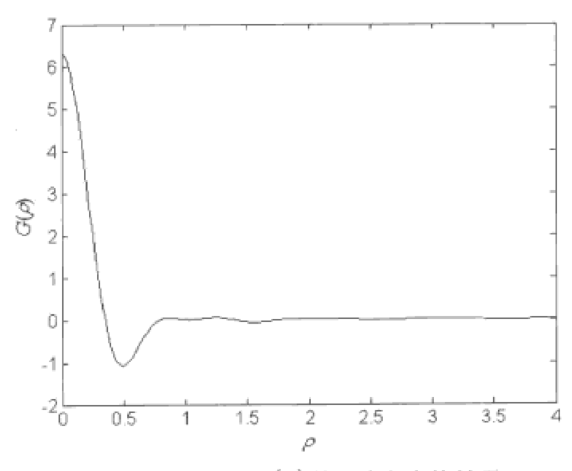


图 14.17　函数$g_1(r)$的汉克尔变换结果

可见频谱函数$G(\rho)$在$\rho=4$附近已经趋于 0，可见频谱是限制在一个有限区域的。在 MathWorks 网站上可以下载到汉克尔变换的一个演示程序：http://www.mathworks.com/matlabcentral/ fileexchange/loadFile.do?objectId=6570。

执行下载文件 Hankel_transform.m 可以得到如图 14.18 所示的结果。

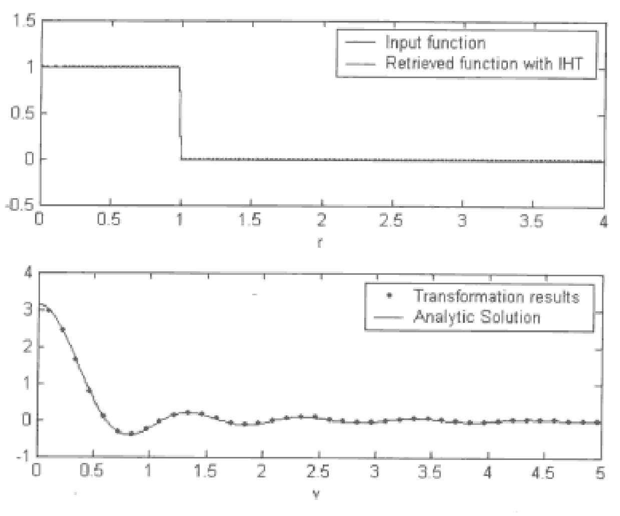


图 14.18　阶越函数的汉克尔变换结果

下面网址可以下载到计算零阶和一阶汉克尔变换的程序：http://infohost.nmt.edu/~borchers/hankel.html。

下面网址可以下载到计算离散汉克尔变换的程序包：http://www.mathworks.de/ matlabcentral/fileexchange/loadFile.do?objectId=13371&objectType=file。

离散汉克尔函数 dht 的调用格式为：

```
[H,k,r,I,K,R0,h0]=dht(h,R,N,n);
```

参数说明： H 是输出的变换结果。k 是频谱的空间频率。r 是半径位置。I 是积分核，是一个矩阵。K 是谱因子。R0 是信号因子。h0 是离散信号。H 是输入函数的句柄。R 是最大半径。N 是信号长度。N 是变换的阶数。

例 14–15： 利用 dht 函数计算下面函数的零阶离散汉克尔变换，其中采样点数目是 200。

$$g_2(x)=\begin{cases}1, & 0\le x\le 1\\ 0, & 1<x\le 4\end{cases}$$

分析： 利用函数 linspace 来生成离散采样点，最大半径是 4，即 R=4。变换的输入参数 N=200，n=0。在上面已知的条件下可调用函数 dht 来计算零阶离散汉克尔变换。函数 $g_2(x)$ 需要使用一个函数文件来定义，即：

```
function h=testfun(t);
h=zeros(size(t));
h(t<=1)=1;     % 计算函数值
```

调用下面的程序可以计算汉克尔变换：

```
R=4;    % 对参数 R 赋值
N=200;  % 对参数 N 赋值
n=0;    % 对参数 n 赋值
[H,k,r,I,K,R,h]=dht('testfun',R,N,n); % 计算离散汉克尔变换
plot(r,H,'k'); % 绘图
```

计算结果如图 14.19 所示。该变换与离散余弦结果很相似，较大数主要分布在 0 附近。

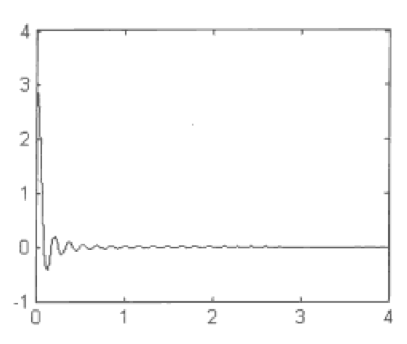


图 14.19　离散汉克尔变换的结果

14.7　小波变换

对于分析和处理非平稳信号，傅里叶变换在解决这类问题受到限制。为此，国内外研究者在近几十年做了不断探索，发展了不同形式的变换，如窗口傅里叶变换、分数傅里叶变换以及小波变换等。本节介绍小波变换。

令函数 $\psi(t)$ 是平方可积的实数空间（能量有限的信号空间），函数 $\psi(t)$ 的傅里叶变换是 $\Psi(\omega)$。当 $\Psi(\omega)$ 满足允许条件（Admissible Condition）：$AC=\int_R \frac{|\Psi(\omega)|^2}{|\omega|}\mathrm{d}\omega<M$，$M$ 是一个有限的正实数，函数 $\psi(t)$ 就可以作为一个基小波（或者叫做母小波，Mother Wavelet）。把基小波 $\psi(t)$ 平移和伸缩后就得到了小波序列。对于连续情况，小波序列可以写为：

$$\psi_{a,b}(t)=\frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right) \tag{14-38}$$

其中 $a\ (a\neq 0)$ 和 b 分别是伸缩因子和平移因子，它们都是实数。对于离散的情况有：

$$\psi_{z,s}(t)=2^{-z/2}\psi\left(2^{-z}t-s\right) \tag{14-39}$$

其中 $z(z\neq 0)$ 和 s 分别是伸缩和平移因子，它们是整数。对于任意连续函数 $f(t)$ 的小波变换可以表示为：

$$W_f(a,b)=\langle f,\psi_{a,b}\rangle=|a|^{-1/2}\int_R f(t)\overline{\psi\left(\frac{t-b}{a}\right)}\mathrm{d}t \tag{14-40}$$

其中 $\langle f,\psi_{a,b}\rangle$ 表示内积，$\overline{\psi\left(\frac{t-b}{a}\right)}$ 表示取复共轭。逆小波变换为：

$$f(t)=\frac{1}{AC}\iint\frac{1}{a^2}W_f(a,b)\psi\left(\frac{t-b}{a}\right)\mathrm{d}a\mathrm{d}b \tag{14-41}$$

下面来介绍小波函数的相关知识。

14.7.1　管理小波函数

函数 wavemngr 可以来管理小波函数，其调用格式为：

```
wavemngr('add',fn,fsn,wt,nums,file)                %格式 1
wavemngr('add',fn,fsn,wt,nums,file,b)              %格式 2
wavemngr('add',fn,fsn,wt,{nums,typnums},file)      %格式 3
wavemngr('add',fn,fsn,wt,{nums,typnums},file,b)    %格式 4
wavemngr('del',n)                                  %格式 5
wavemngr('restore')                                %格式 6
wavemngr('restore',in2)                            %格式 7
wavemngr('read')                                   %格式 8
wavemngr('read',in2)                               %格式 9
wavemngr('read_asc')                               %格式 10
```

参数说明：格式 1 至格式 4 用来添加小波函数。fn 是 Family name，即小波族。Fsn 是 Family short name，即小波族的缩写名。wt 是 Wavelet type 的缩写，即小波类型，当 wt=1 时为正交小波；当 wt=2 时为双正交小波；当 wt=3 时为带尺度函数的小波；当 wt=4 时为不带尺度函数的小波；当 wt=5 时为不带尺度函数的复小波。nums 为字符串的个数。file 是小波函数名。b 为小波有效支撑的上下界。typnums 用于指定小波参数，其取值为 integer，real，string，其中默认值为 integer。格式 5 用于删除小波函数，n 是小波函数名。格式 6 用于保存前面的小波函数，格式 7 保存最初小波函数。格式 8 至格式 10 用来读取小波函数。格式 8 用来读取小波族，格式 9 用来读取小波函数名，格式 10 用来读取小波函数的详细信息。

比如用户在命令窗中输入 wavemngr('read')，将会出现下面的信息：

```
ans =
Haar                    haar
Daubechies              db
Symlets                 sym
Coiflets                coif
BiorSplines             bior
ReverseBior             rbio
Meyer                   meyr
DMeyer                  dmey
Gaussian                gaus
Mexican_hat             mexh
Morlet                  morl
Complex Gaussian        cgau
Shannon                 shan
Frequency B-Spline      fbsp
Complex Morlet          cmor
```

14.7.2 计算一维小波变换

函数 dwt 可用来计算一维小波变换，其调用格式为：

```
[CA,CD] = dwt(x,'wname');      %格式 1
[CA,CD] = dwt(x,Lo_D,Hi_D);   %格式 2
```

参数说明：x 是输入的一维离散信号。wname 是离散小波变换中基小波名称。Lo_D 和 Hi_D 分别为低频和高频分解滤波器。参数 CA 和 CD 分别是输出的低频系数和高频系数，在格式 1 中它们的长度相等并且等于 ceil(length(x)/2)，在格式 2 中元素数目关系是 length(CA)=length(CD)=floor((length(x)+length(Lo_D)-2)/2)。

例 14-16：MATLAB 中自带离散信号（noisbump.mat 文件对应的数据）的离散小波变换。

分析：利用 load 函数可以读入该数据文件，然后调用 dwt 函数来计算小波变换，这里考虑使用两种格式计算小波变换。最后把计算结果分别显示出来。根据上面的分析可以写出相应的计算程序，具体内容如下：

```
x=importdata('noisbump.mat');      % 读入数据
[CA1,CD1]=dwt(x,'coif4');          % 进行小波变换
[Lo_D,Hi_D]=wfilters('sym2','d');    % 生成低、高频率波向量
[Lo_D,Hi_D,Lo_R,Hi_R]=wfilters('sym2');  % 生成低、高频率波向量
[CA2,CD2]=dwt(x,Lo_D,Hi_D);          % 进行小波变换
a1=subplot(311);
plot(x,'k');xlim([1,length(x)]);              % 画出输入信号
xlabel('(a)','Fontsize',14);
a2=subplot(323);
plot(CA1,'k');xlim([1,length(CA1)]);  % 画出小波变换结果
xlabel('(b)','Fontsize',14);
a3=subplot(324);
plot(CD1,'k');xlim([1,length(CD1)]);  % 画出小波变换结果
xlabel('(c)','Fontsize',14);
a4=subplot(325);
plot(CA2,'k');xlim([1,length(CA2)]);  % 画出小波变换结果
xlabel('(d)','Fontsize',14);
a5=subplot(326);
plot(CD2,'k');xlim([1,length(CD2)]);  % 画出小波变换结果
xlabel('(e)','Fontsize',14);
set([a1,a2,a3,a4,a5],'Fontsize',12);  % 设置字体大小
```

上述程序所得结果如图 14.20 所示。

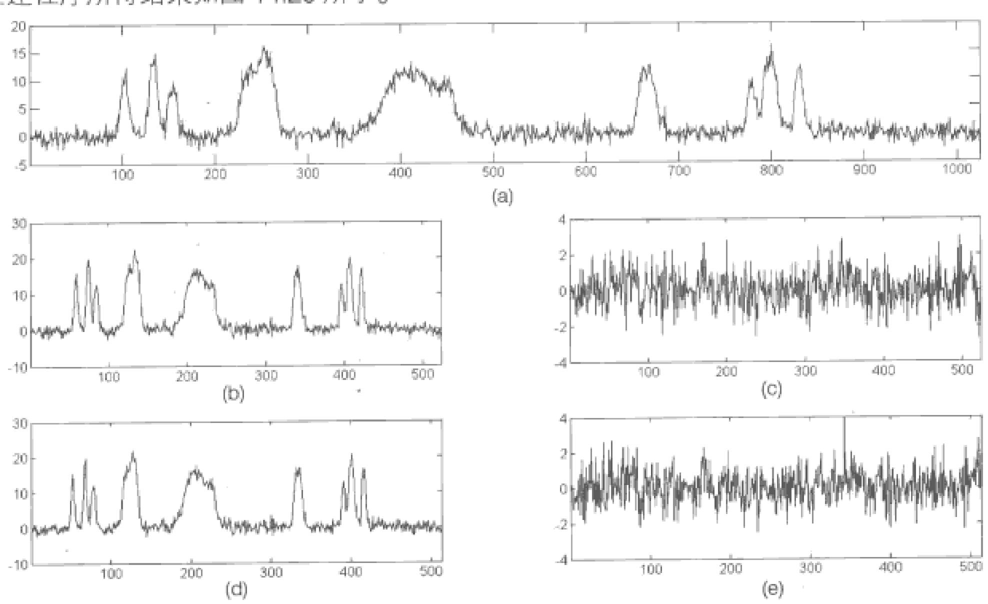


图 14.20　一维小波变换结果

14.7.3 实现逆离散小波变换

函数 idwt 实现逆离散小波变换，其调用格式为：

```
x = idwt(CA,CD,'wname');        %格式 1
x = idwt(CA,CD,Lo_R,Hi_R);      %格式 2
```

参数说明： x 是逆变换所得结果。参数 CA 和 CD 分别是输出的低频系数和高频系数。wname 是离散小波变换中基小波名称。Lo_R 和 Hi_R 分别为低频和高频分解滤波器。格式 1 和格式 2 对应于函数 dwt 的两种调用格式。

例 14-17： 计算逆小波变换。

分析： 这里使用不同基小波和低、高频率波系数进行逆小波变换，比较不同参数对于正/逆小波变换的影响，其中基小波使用 coif4 和 haar 分别测试。

```
ix1=idwt(CA1,CD1,'coif4');    % 用 coif4 基小波进行逆变换
ix2=idwt(CA1,CD1,'haar');     % 用 haar 基小波进行逆变换
[Lo_D2,Hi_D2,Lo_R2,Hi_R2]=wfilters('haar');  % 生成另外一对低、高频率波向量
ix3=idwt(CA2,CD2,Lo_R,Hi_R);              % 使用正变换对应的低、高频率波向量
ix4=idwt(CA2,CD2,Lo_R2,Hi_R2);            % 使用新生成的低、高频率波向量
subplot(411);plot(ix1,'k');hold
on;plot(x,'r');xlim([1,length(ix1)]);xlabel('(a)','Fontsize',12);% 绘图
subplot(412);plot(ix2,'k');hold
on;plot(x,'r');xlim([1,length(ix2)]);xlabel('(b)','Fontsize',12);% 绘图
subplot(413);plot(ix3,'k');hold
on;plot(x,'r');xlim([1,length(ix3)]);xlabel('(c)','Fontsize',12);% 绘图
subplot(414);plot(ix4,'k');hold
on;plot(x,'r');xlim([1,length(ix4)]);xlabel('(d)','Fontsize',12);% 绘图
```

输出图形如图 14.21 所示。

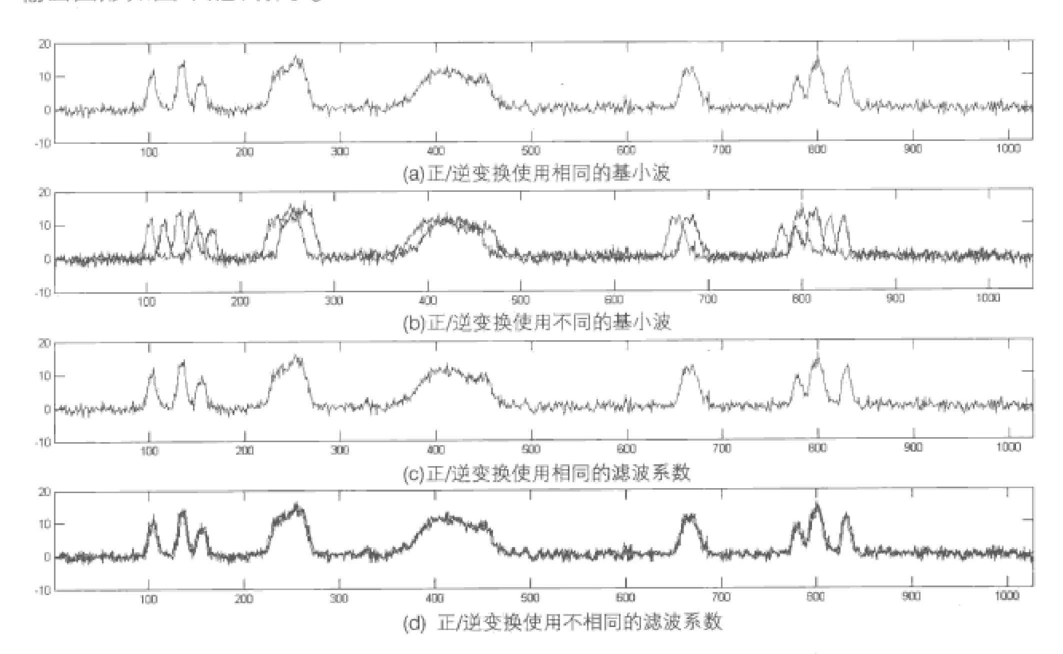


图 14.21 逆小波变换结果



在图 14.21 的图(a)和图(c)中两条曲线重合，说明正逆变换对应；而图(b)和图(d)出现了不重合的现象，即正逆变换不是对应的。

14.7.4 实现二维离散小波变换

函数 dwt2 可以实现二维离散小波变换，其调用格式为：

```
[CA,CH,CV,CD] = dwt2(x,'wname');    %格式 1
[CA,CH,CV,CD] = dwt2(x,Lo_D,Hi_D); %格式 2
```

参数说明：CA，CH，CV，CD 分别为低频系数、水平高频系数、竖直高频系数和对角线高频系数。x 是输入图像对应的矩阵。wname 是基小波名称。Lo_D 和 Hi_D 分别是低、高频分解滤波系数。

例 14–18：计算如图 14.22 所示的图像的小波变换。

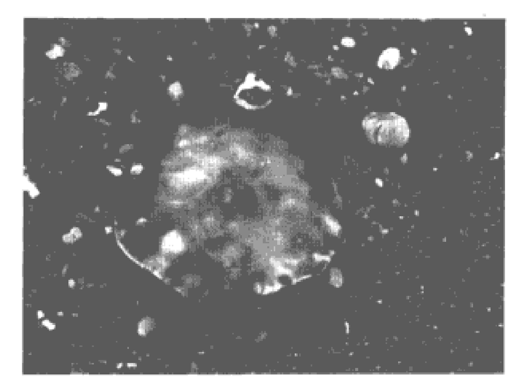


图 14.22　输入图像



图 14.22 对应的数据存放于 MATLAB 安装路径下\MATLAB\R2008a\toolbox\wavelet\wavedemo 的 jellyfish.mat 文件。

分析：首先利用函数 importdata 或者 load 函数读入图像对应的数据，然后调用函数 dwt2 进行小波变换，用户可以根据需要选择不同的基小波函数，这里采用 haar 基小波进行小波变换。实现程序如下：

```
[CA,CH,CV,CD] = dwt2(X,'haar');       % 进行二维小波变换
figure;
subplot(221);imshow(CA,[]);xlabel('CA'); % 显示低频分量
subplot(222);imshow(CH,[]);xlabel('CH'); % 显示水平高频分量
subplot(223);imshow(CV,[]);xlabel('CV'); % 显示竖直高频分量
subplot(224);imshow(CD,[]);xlabel('CD'); % 显示对角线高频分量
```

计算结果如图 14.23 所示，可见图像的二维小波主要信息集中在低频部分。

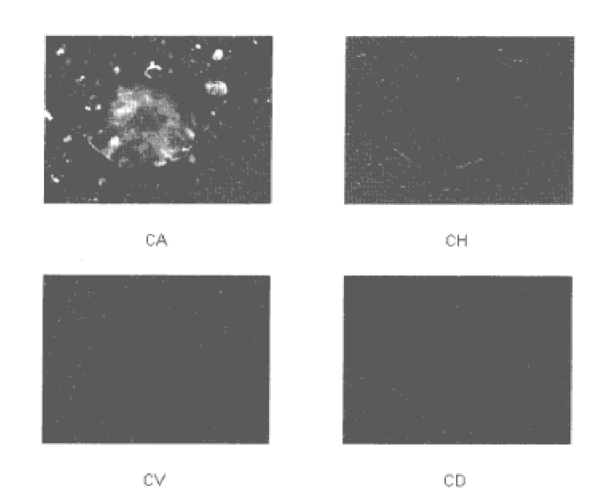


图 14.23 二维小波变换结果

14.7.5 实现二维逆小波变换

函数 idwt2 可以实现二维逆小波变换，其调用格式为：

```
X = idwt2(CA,CH,CV,CD,'wname');      %格式 1
X = idwt2(CA,CH,CV,CD,Lo_R,Hi_R);    %格式 2
```

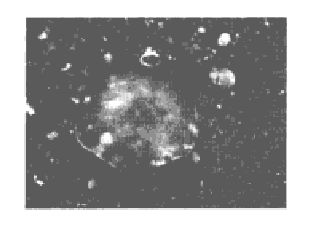
参数说明： X 是返回的逆变换结果。CA，CH，CV，CD 分别为低频系数、水平高频系数、竖直高频系数和对角线高频系数。wname 是基小波名称。Lo_R 和 Hi_R 分别是低频、高频分解滤波系数。

例 14–19： 二维逆小波变换。

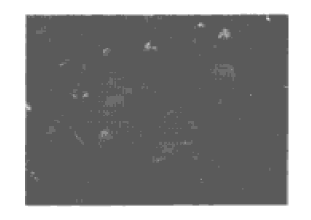
分析： 与一维逆变换类似，这里使用两种基小波进行逆变换测试，所用基小波为 haar 小波和 db3 小波，同时显示其逆变换结果并进行比较。计算程序如下：

```
X1 = idwt2(CA,CH,CV,CD,'haar'); % 利用 haar 小波进行逆变换
X2 = idwt2(CA,CH,CV,CD,'db3'); % 利用 db3 小波进行逆变换
figure;
subplot(121);imshow(X1,[]);xlabel('(a)','Fontsize',14); % 绘图
subplot(122);imshow(X2,[]);xlabel('(b)','Fontsize',14); % 绘图
```

上述程序输出结果如图 14.24 所示。



(a)利用 haar 小波



(b)利用 db3 小波

图 14.24 二维逆小波变换结果

从图 14.24 可见图(a)和图(b)大体相似，但是在细节和亮度上存在差异，其中图(a)准确地回到

了原输入图像。

14.8 小结

本章主要介绍了几种变换的程序实现，即分数傅里叶变换、菲涅尔变换、Hartley 变换、离散正/余弦变换、分数随机变换、汉克尔变换以及小波变换等。傅里叶变换是这些变换的基础。分数傅里叶变换可以根据快速傅里叶变换算法来计算，因而具有较快的速度。菲涅尔变换对应于菲涅尔衍射，可以表示为卷积形式，因此可调用 FFT 算法计算。离散正/余弦变换可以看成将离散傅里叶变换的实部和虚部两部分独立出来，它们都是实变换，特别是离散余弦变换在信号和图像处理中有着广泛的应用。分数随机变换是使输出具有随机性的变换，同时这个变换还可以保持一些好的数学性质，是一种新的变换工具。汉克尔变换可以用来解决圆对称函数的问题。小波变换是一种新兴的数学工具，目前已经在多个科学领域应用。

◆ ◆ ◆

第15章 特殊函数

本章包括

- **Bessel 函数** 介绍5种贝塞尔函数及艾里函数。
- **Hermite 函数** 介绍 Hermite 函数及 Hermite-Gaussian 函数。
- **阶乘函数与 Gamma 函数** 给出阶乘的计算方法和 Gamma 函数的计算。
- **Beta 函数** 介绍 Beta 函数及不完全 Beta 函数的数值计算。
- **其他特殊数学函数** 介绍雅可比椭圆函数、椭圆函数、误差函数等。

本章主要介绍在数学、物理等学科中常用到的特殊函数的定义及相应的 MATLAB 计算函数，如贝塞尔函数、厄尔米特函数、伽玛函数等。利用这些特殊函数可以简练地表达计算过程中的结果，了解它们的计算方法，促进特殊问题的研究。

15.1 Bessel 函数

贝塞尔函数存在 5 种形式，即第一类贝塞尔函数、第二类贝塞尔函数、第一类修正的贝塞尔函数、第二类修正的贝塞尔函数以及第三类贝塞尔函数。第一类贝塞尔函数 $J_v^1(z)$ 的数学定义为：

$$J_v^1(z)=\left(\frac{z}{2}\right)^v\sum_{k=0}^{\infty}\frac{\left(-\frac{z^2}{4}\right)^k}{k!\Gamma(v+k+1)} \tag{15-1}$$

其中 v 是第一类贝塞尔函数的阶数。这是一个无穷级数，MATLAB 提供了函数 besselj 来计算第一类贝塞尔函数。函数 besselj 的调用格式为：

```
J = besselj(nu,z);
```

参数说明： J 是输出的第一类贝塞尔函数值。nu 是贝塞尔函数的阶数。z 是离散的采样点。其中参数 nu 可以是一个向量。当 nu 是行向量时，z 要求是列向量；当 nu 是列向量时，z 要求是行向量，这时输出的 J 是一个矩阵。

下面这段程序用来计算 0 到 4 阶贝塞尔函数的曲线。

```
z=linspace(0,12,400);         % 自变量离散采样
J1=besselj([0:4]',z);           % 计算第一类贝塞尔函数值
plot(z,J1);                 % 绘图
xlabel('{\itz}','Fontname','Times new roman','Fontsize',14);  % 标注
ylabel('{\itJ}_{\itv}({\itz})','Fontname','Times new roman','Fontsize',14);  %
标注
gtext('{\itv}=0','Fontsize',14,'Fontname','Times new roman');  % 标注
gtext('{\itv}=1','Fontsize',14,'Fontname','Times new roman');  % 标注
gtext('{\itv}=2','Fontsize',14,'Fontname','Times new roman');  % 标注
gtext('{\itv}=3','Fontsize',14,'Fontname','Times new roman');  % 标注
```

```
gtext('{\itv}=4','Fontsize',14,'Fontname','Times new roman');  % 标注
```



这里使用了函数 gtext 来标注，用户可以用鼠标在相应位置加标注。

上述程序输出结果如图 15.1 所示，0 阶贝塞尔函数在 0 点处等于 1，而其他阶的贝塞尔函数等于 0，同时它们随着 z 的增加振幅变小。

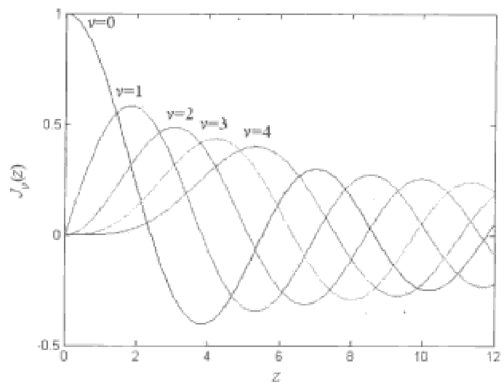


图 15.1　第一类贝塞尔函数曲线图

第二类贝塞尔函数 $Y_v^2(z)$ 的数学定义为：

$$Y_v^2(z)=\frac{J_v^1(z)\cos(v\pi)-J_{-v}^1(z)}{\sin(v\pi)} \tag{15-2}$$

其中 v 为实数，第二类贝塞尔函数 $Y_v^2(z)$ 由第一类贝塞尔线性函数组合而成。

MATLAB 函数 bessely 用来计算第二类贝塞尔函数，其调用格式为：

```
Y = bessely(nu,z);
```

参数说明： Y 是输出的第二类贝塞尔函数值。nu 是贝塞尔函数的阶数。z 是离散的采样点。其中参数 nu 可以是一个向量。当 nu 是行向量时，z 要求是列向量；当 nu 是列向量时，z 要求是行向量，这时输出的 Y 是一个矩阵。

下面一段程序用来计算第二类贝塞尔函数值。

```
z=linspace(1,6,400);                % 自变量离散采样
Y1=bessely([0:2]'+0.05,z);        % 计算第二类贝塞尔函数值
plot(z,Y1);                        % 绘图
Y11=bessely([0:2]'+0.05,1)
xlabel('{\itz}','Fontname','Times new roman','Fontsize',14);              % 标注
ylabel('{\itY}_{\itv}({\itz})','Fontname','Times new roman','Fontsize',14);   %
标注
```

上述程序输出结果如图 15.2 所示，与第一类贝塞尔函数相比，第二类贝塞尔函数变换振荡不是那么剧烈。

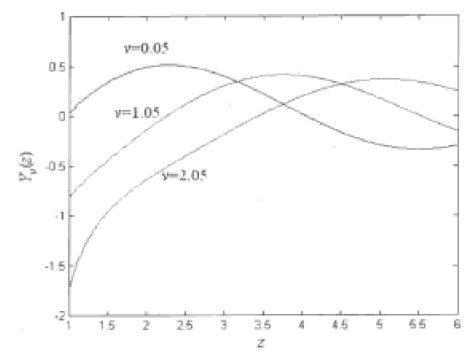


图 15.2 第二类贝塞尔函数的曲线

第一类修正的贝塞尔函数 $I_\nu^1(z)$ 的数学定义为：

$$I_\nu^1(z)=\left(\frac{z}{2}\right)^\nu\sum_{k=0}^{\infty}\frac{\left(\frac{z^2}{4}\right)^k}{k!\Gamma(\nu+k+1)} \tag{15-3}$$

可见 $I_\nu^1(z)$ 与 $J_\nu^1(z)$ 相差不多。

MATLAB 函数 besseli 可以计算第一类修正的贝塞尔函数，其调用格式为：

```
I = besseli(nu,z);
```

参数说明： I 是第一类修正的贝塞尔函数值。nu 是贝塞尔函数的阶数。z 是离散的采样点。其中参数 nu 可以是一个向量。当 nu 是行向量时，z 要求是列向量；当 nu 是列向量时，z 要求是行向量，这时输出的 I 是一个矩阵。

下面一段程序用来计算第一类修正的贝塞尔函数值。

```
z=linspace(0,2,400);   % 自变量离散采样
I1=besseli([0:3]',z);    % 计算第一类修正的贝塞尔函数值
plot(z,I1);          % 绘图
I11=besseli([0:3]',1)    % 计算特殊点的值
xlabel('{\itz}','Fontname','Times new roman','Fontsize',14);            % 标注
ylabel('{\itI}_{\itv}({\itz})','Fontname','Times new roman','Fontsize',14);  %
标注
```

输出结果如图 15.3 所示，从图中可以看出第一类修正的贝塞尔函数值的绝对值能够超过 1。

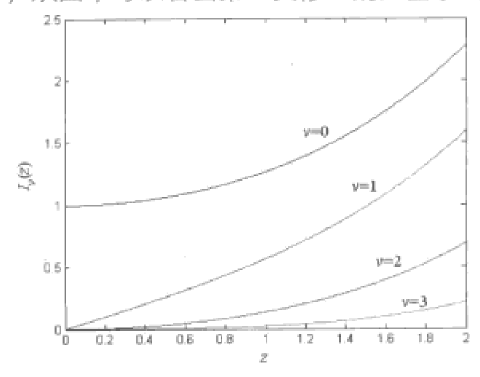


图 15.3 第一类修正的贝塞尔函数曲线

第二类修正的贝塞尔函数 $K_v^2(z)$ 的数学定义如下：

$$K_v^2(z)=\left(\frac{\pi}{2}\right)\frac{I_{-v}^2(z)-I_v^2(z)}{\sin(v\pi)} \tag{15-4}$$

公式（15-4）中 v 不能取整数。

MATLAB 函数 besselk 可以计算第二类修正的贝塞尔函数，其调用格式为：

```
K = besselk(nu,z);
```

参数说明： K 是第二类修正的贝塞尔函数的输出。nu 是贝塞尔函数的阶数。z 是离散的采样点。其中参数 nu 可以是 1 个向量。当 nu 是行向量时，z 要求是列向量；当 nu 是列向量时，z 要求是行向量，这时输出的 K 是一个矩阵。

下面程序是用来计算第二类修正贝塞尔函数的例子。

```
z=linspace(0,3,400);       % 自变量离散采样
K1=besseli([0:3]'+0.5,z);   % 计算第一类修正的贝塞尔函数值
plot(z,K1);                % 绘图
K11=besseliki([0:3]',1)      % 计算特殊点的值
xlabel('{\itz}','Fontname','Times new roman','Fontsize',14);           % 标注
ylabel('{\itK}_{\itv}({\itz})','Fontname','Times new roman','Fontsize',14);  %
标注
```

上述程序计算结果如图 15.4 所示，可见第二类修正的贝塞尔函数很快地增加并超过 1。

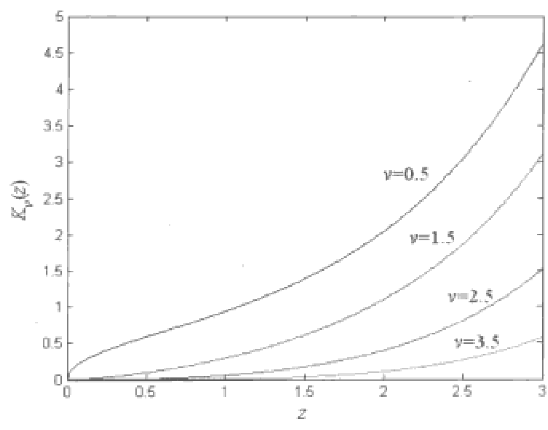


图 15.4　第二类修正的贝塞尔函数

第三类贝塞尔函数（也被称为汉克尔函数）$H_v^k(z)$ 的数学定义为：

$$H_v^k(z)=J_v^1(z)+(-1)^{k+1}\,\mathrm{i}Y_v^2(z) \tag{15-5}$$

其中 v 是贝塞尔函数的阶数，$k=1,2$，i 是虚数单位。

MATLAB 函数 besselh 用来计算第三类贝塞尔函数，其调用格式为：

```
H = besselh(nu,k,z);
```

参数说明： H 是第三类贝塞尔函数的输出结果。nu 是贝塞尔函数的阶数。z 是离散的采样点。

其中参数 nu 可以是一个向量。当 nu 是行向量时，z 要求是列向量；当 nu 是列向量时，z 要求是行向量，这时输出的 H 是一个矩阵。

下面一段程序是计算第三类贝塞尔函数的例子。

```
z=linspace(1,3,400);                % 自变量离散采样
H1=besselh([0:2]'+0.3,1,z);        % 计算第三类贝塞尔函数值
subplot(121);
plot(z,real(H1));                   % 绘制实部
xlabel('{\itz}','Fontname','Times new roman','Fontsize',14);          % 标注
ylabel('real[{\itH}_{\itv}({\itz})]','Fontname','Times new roman','Fontsize',14);   % 标注
axis square;
subplot(122);
plot(z,imag(H1),':');               % 绘制虚部
axis square;
K11=besselH([0:2]',1)              % 计算特殊点的值
xlabel('{\itz}','Fontname','Times new roman','Fontsize',14);          % 标注
ylabel('imag[{\itH}_{\itv}({\itz})]','Fontname','Times new roman','Fontsize',14);   % 标注
```

上述程序计算后得到的结果如图 15.5 所示，这里给出了第三类贝塞尔函数实部和虚部变化情况。

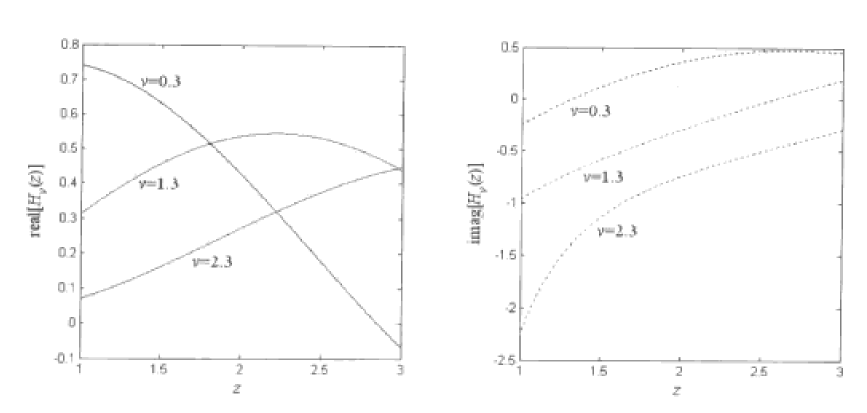


图 15.5　第三类贝塞尔函数曲线（左图是实部、右图是虚部）

艾里（airy）函数是与修正的贝塞尔函数有关的函数，其数学描述为：

$$A(z)=\left[\frac{1}{\pi}\sqrt{z/3}\right]K_{1/3}^{2}(\zeta),\quad B(z)=\sqrt{z/3}\left[I_{-1/3}^{2}(\zeta)+I_{1/3}^{2}(\zeta)\right],\quad \zeta=\frac{2}{3}z^{3/2} \qquad (15\text{-}6)$$

其中 $A(z)$ 为第一类艾里函数，$B(z)$ 为第二类艾里函数。

MATLAB 提供函数 airy 来计算艾里函数，其调用格式为：

```
W = airy(z);
W = airy(k,z);
```

参数说明： W 是输出的艾里函数值。z 是自变量采样值。k 用来指定艾里函数分类，其值为 0 对应于第一类艾里函数，为 1 对应于派生的第一类艾里函数，为 2 对应于第二类艾里函数，为 3

对应于派生的第二类艾里函数，k 的默认值为 1。

下面一段程序用来计算艾里函数：

```
z=linspace(0,2,201);       % 自变量离散采样
for k=0:3;
    subplot(1,4,k+1);
    W=airy(k,z);            % 计算艾里函数值
    plot(z,real(W),'k');       %绘制实部
    axis square;            % 把坐标轴设为方形
    xlabel({'{\itz}',['(',char(97+k),')']});  % 标注
end
```

上述程序计算结果如图 15.6 所示，可见第一类艾里函数单调递减而另外三类图形单调递增。

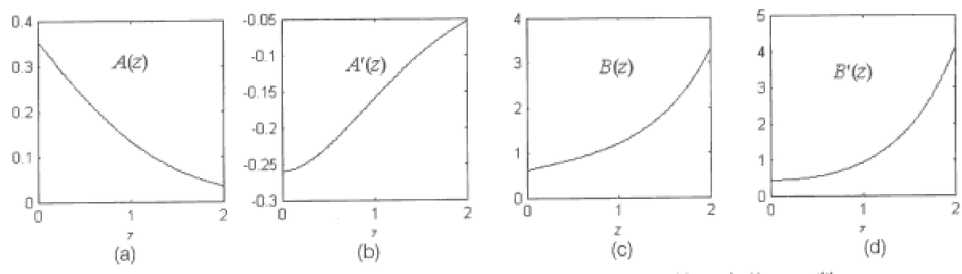


图 15.6　艾里函数图：(a)第一类艾里函数；(b)派生的第一类艾里函数；(c)第二类艾里函数；(d)派生的第二类艾里函数

本节介绍的函数保存于 MATLAB 软件的安装路径下，即\toolbox\matlab\specfun，用户从中可以看到一些计算特殊函数的 M 文件。

15.2　Hermite 函数

厄尔米特多项式函数 $H_n(x)$ 的母函数是：

$$H_n(x)=(-1)^n e^{x^2}\frac{d^n}{dx^n}e^{-x^2} \tag{15-7}$$

不同阶数之间的递推关系是：

$$H_{n+1}(x)-2xH_n(x)+2nH_{n-1}(x)=0 \tag{15-8}$$

特别地，$H_0(x)=1$，$H_1(x)=2x$，利用 $H_0(x)$ 和 $H_1(x)$ 可以递推出其他阶数的厄尔米特多项式函数。

在 MATLAB 中，可以利用 maple 函数来计算厄尔米特函数，在调用 maple 函数计算之前需要加载扩展符号工具箱，即：

```
maple('with','orthopoly')
```



如果用户未安装符号工具箱（Symbolic toolbox）和扩展符号工具箱（Extend symbolic toolbox），MATLAB 将提示用户安装这个工具箱。

执行上述语句将会提示下面信息：

```
ans =[G, H, L, P, T, U]
```

其中字母含义如表 15.1 所示。

表 15.1 正交多项式函数

函数名简名	调用格式	说明
G	maple('G',n,a,x)	格根包尔多项式函数， a 是一个无理代数表达式或者是一个大于-1/2 的有理数
H	maple('H',n,x)	厄尔米特多项式函数
L	maple('L',n,x)	拉盖尔多项式函数
	maple('L',n,a,x)	广义拉盖尔多项式函数，a 是一个无理代数表达式或者是一个大于-1 的有理数
P	maple('P',n,x)	勒让德多项式函数
	maple('P',n,a,b,x)	雅可比多项式函数，a 和 b 是无理代数表达式或者是大于-1 的有理数
T	maple('T',n,x)	第一类切比雪夫多项式函数
U	maple('U',n,x)	第二类切比雪夫多项式函数



n 表示多项式函数的阶数，x 是自变量，a 和 b 是参数。

下面的程序代码用来计算厄尔米特多项式函数值并画出相应曲线。

```
x=linspace(0,5,201);                    % 离散化自变量
for n=2:4;
   for k=1:length(x);
      Hx(k)=eval(maple('H',n,x(k)));      % 计算函数值
   end
   subplot(1,3,n-1);
   plot(x,Hx);                         %绘图
   axis square;                        % 设置坐标轴为方形
end
```

上述程序计算结果如图 15.7 所示，从图中可以看出厄尔米特多项式函数是单调递增函数。

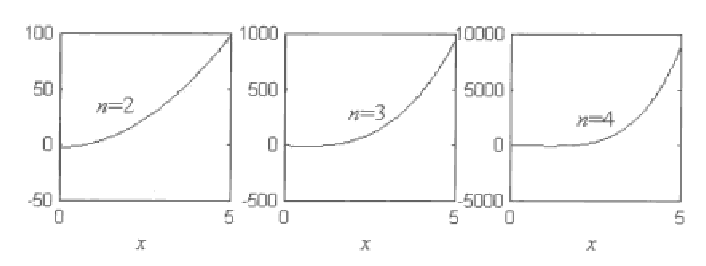


图 15.7 厄尔米特多项式函数对应的曲线

厄尔米特-高斯函数的定义补充如下：

$$\varphi_n(x)=\frac{1}{\sqrt{2^n\sqrt{\pi}n!}}H_n(x)\exp\left(-x^2/2\right) \quad (15\text{-}9)$$

上式中 n 表示阶数。$H_n(x)$是 n 阶厄尔米特多项式函数。厄尔米特-高斯函数是分数傅里叶变换的本征值。

下面利用程序计算出不同阶数 n 下的厄尔米特-高斯函数，程序如下：

```
maple('with','orthopoly')
x=linspace(-3,3,401);                          % 离散化自变量
for n=0:5;
    for k=1:length(x);
        Hx(k)=eval(maple('H',n,x(k)));         % 计算函数值
    end
    subplot(2,3,n+1);
    HG=Hx.*exp(-x.^2/2)/sqrt(2^n*sqrt(pi)*prod(1:n));
    plot(x,HG,'k');                            %绘图
    axis square;                               % 设置坐标轴为方形
    xlabel('{\itx}','Fontsize',12,'Fontname','Times new roman');
    xlim([min(x),max(x)]);                     % 设置坐标轴范围
end
```

运行上述程序，所得结果如图 15.8 所示。根据曲线的对称性，可以知道奇数阶厄尔米特-高斯函数是奇函数，而偶数阶厄尔米特-高斯函数是偶函数。

其他特殊函数，如格根包尔多项式函数、拉盖尔多项式函数、勒让德多项式函数、雅可比多项式函数、切比雪夫多项式函数等可以类似于上面计算厄尔米特多项式函数的例子进行计算。

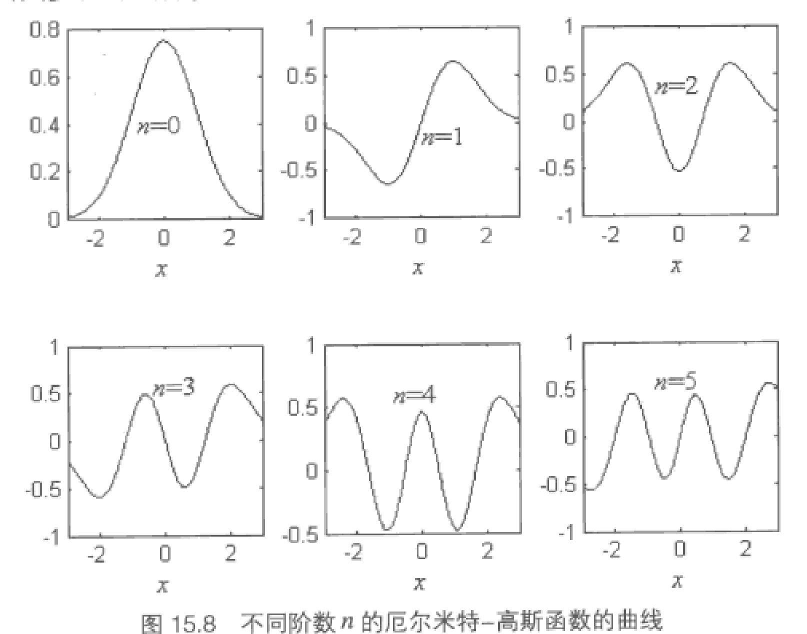


图 15.8　不同阶数 n 的厄尔米特-高斯函数的曲线

15.3　阶乘函数与 Gamma 函数

在 MATLAB 中，阶乘的计算有多种方式，其中函数 factorial 是专门用来计算阶乘的，其调用格式为：

```
f = factorial(N);
```

参数说明： f是返回的阶乘结果。N是输入参数，其要求是一个非负整数。

比如用户在命令窗中输入：

```
>> factorial(4)
```

将会返回：

```
ans =24
```

还可以用下面的方式来计算阶乘，如：

```
f = prod(1:N);
```

用户在命令窗中输入：

```
>> prod(1:3),prod(1:0)
```

执行后会得到：

```
ans =     6
ans =     1
```

这里利用函数prod计算得出0的阶乘等于1，当N=0时，1:N是一个空矩阵，即空矩阵的连乘积等于1。

还可以用递归算法来计算阶乘，具体实现程序的内容如下：

```
function f= facto(n);    % 利用递归算法计算阶乘的函数
if n==0;
    f=1;
elseif n>0.5
    f=n*facto(n-1);
end
```

参数说明： f和n分别是函数facto的输入和输出参数。这个函数的用法和factorial相似。

调用函数facto来计算5!可以用下面的格式：

```
>> f= facto(5)
```

输出为：

```
f =   120
```

Gamma函数$\Gamma(a)$的数学定义为：

$$\Gamma(a)=\int_0^\infty \mathrm{e}^{-t}t^{a-1}\mathrm{d}t \tag{15-10}$$

特别地，当a是整数（即$a=n$）时，有关系式$\Gamma(n+1)=n!$成立。

在MATLAB中，Gamma函数的计算用函数gamma来实现。其调用格式为：

```
y = gamma(x);
```

参数说明： y 和 x 分别是函数 gamma 的输出和输入。其中参数 $x \le 0$ 时返回的结果是 inf。

下面通过计算 Gamma 函数的曲线来比较 $\Gamma(n+1)=n!$ 的关系。

```
x=linspace(0.1,4,1601);  % 对自变量离散采样
plot(x,gamma(x)); % 绘制 Gamma 函数对应的曲线
hold on;
plot(1:4,factorial(0:3),'x','markersize',12);
set(gca,'Fontsize',14);          % 设置坐标轴字体
xlabel('{\ita}','Fontsize',22,'Fontname',…
'Times new roman');  %标注
ylabel('\Gamma({\ita})','Fontsize',22,'Fontname',…
'Times new roman');      %标注
set(gcf,'Color','w');            %设置背景色为白色
```

所得图形如图 15.9 所示，可见 $\Gamma(n+1)$ 和 $n!$ 对应数值吻合得很好。

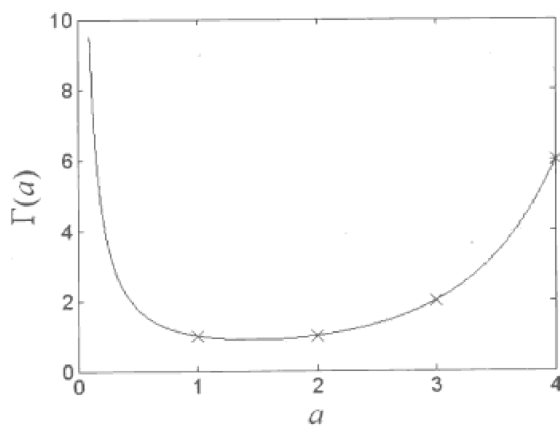


图 15.9　Gamma 函数曲线

不完全 Gamma 函数 $P_1(x)$ 和 $P_2(x)$ 的数学定义如下：

$$P_1(x,a)=\frac{1}{\Gamma(a)}\int_0^x e^{-t}t^{a-1}dt \tag{15-11}$$

$$P_2(x,a)=\frac{1}{\Gamma(a)}\int_x^{\infty} e^{-t}t^{a-1}dt=1-P_1(x) \tag{15-12}$$

根据式（15-11）和（15-12）可知，$0\le P_1(x,a),P_2(x,a)\le 1$。上面函数 $P_1(x)$ 和 $P_2(x)$ 的计算可以由 MATLAB 函数 gammainc 来实现，该函数的调用格式为：

```
Y = gammainc(x, a);
Y = gammainc(x, a, tail);
```

参数说明： Y 是不完全 Gamma 函数的计算结果。x 是不完全 Gamma 函数的积分上限或者下限。a 是不完全 Gamma 函数的阶数。参数 tail 用于指定计算函数 $P_1(x)$ 还是 $P_2(x)$，其中参数 lower 对应于 $P_1(x)$，upper 对应于 $P_2(x)$，lower 是默认值。

下面利用函数 gammainc 来计算不完全 Gamma 函数 $P_1(x)$ 和 $P_2(x)$ 的曲线。根据前面介绍的函数用法可以写出相应的程序，其内容如下：

```
x=linspace(0,8,401);          % 自变量 x 的离散采样
y1=gammainc(x,1);            % 计算不完全 Gamma 函数
y1u=gammainc(x,1,'upper');   % 计算不完全 Gamma 函数
y2=gammainc(x,2);            % 计算不完全 Gamma 函数
y2u=gammainc(x,2,'upper');   % 计算不完全 Gamma 函数
S1=subplot(121);plot(x,y1);hold on;plot(x,y1u,'k:');axis square;   % 绘图
L1=legend('{\itP}_1({\itx},1)','{\itP}_2({\itx},1)',0);              % 标注线型
xlabel('{\itx}','Fontsize',22,'Fontname','Times new roman');        % X 轴标注
ylabel('{\itP}_1({\itx},1), {\itP}_2({\itx},1)','Fontsize',22,'Fontname','Times
new roman');  % Y 轴标注
S2=subplot(122);plot(x,y2);hold on;plot(x,y2u,'k:');axis square;    % 绘图
L2=legend('{\itP}_1({\itx},2)','{\itP}_2({\itx},2)',0);              % 标注线型
xlabel('{\itx}','Fontsize',22,'Fontname','Times new roman');         % X 轴标注
ylabel('{\itP}_1({\itx},2), {\itP}_2({\itx},2)','Fontsize',22,'Fontname','Times
new roman');  % Y 轴标注
set([L1,L2],'Fontsize',14,'Fontname','Times new roman');
set([S1,S2],'Fontsize',14);                                    % 设置坐标轴字体
```

所得图形如图 15.10 所示，参数 a 不同时不完全 Gamma 函数的趋势大体相同。

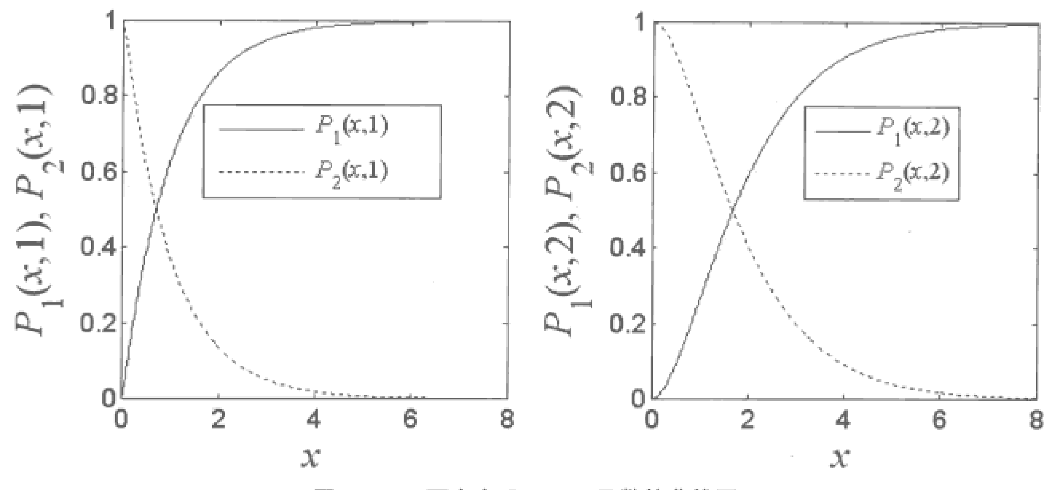


图 15.10　不完全 Gamma 函数的曲线图

函数 gammaln 可以用来计算 Gamma 函数的自然对数，其调用格式为：

```
y = gammaln(a)
```

参数说明： y 和 a 分别是函数 gammaln 的输出和输入。这个函数在功能上等价于 log(gamma(a))。

15.4　Beta 函数

Beta 函数 $B(z,w)$ 的数学定义是：

$$B(z,w)=\int_0^1 t^{z-1}(1-t)^{w-1}=\frac{\Gamma(z)\Gamma(w)}{\Gamma(z+w)} \tag{15-13}$$

当 z 和 w 取整数（即 $z=j=n-k$， $w=k$ ）时，Beta 函数可以用来计算组合数 C_{n-2}^{j-1}，如：

$$B(j,k)=\frac{\Gamma(j)\Gamma(k)}{\Gamma(j+k)}=\frac{\Gamma(j-1+1)\Gamma(k-1+1)}{\Gamma(j+k-1+1)}=\frac{(j-1)!(k-1)!}{(j+k-1)!}=\frac{(j-1)!(k-1)!}{(j+k-2)!}\frac{(j+k-2)!}{(j+k-1)!}$$
$$=\frac{\Gamma(j+k-1)}{\Gamma(j+k)C_{j+k-2}^{j-1}} \tag{15-14}$$

在 MATLAB 中，函数 beta 可以用来计算 Beta 函数，其调用格式为：

```
B = beta(z,w);
```

参数说明： B 是返回的 beta 函数值。z 和 w 分别是 beta 函数两个独立的自变量。

相应的函数 betaln 可以用来计算 Beta 函数的自然对数，其调用格式为：

```
B = betaln(z,w);
```

参数说明： 参数 B，z 和 w 的意义同函数 Beta 中的说明。betaln(z,w)等价于 log(beta(z,w))。

组合数 C_n^k 的计算可以使用函数 nchoosek 来计算，其调用格式为：

```
v = nchoosek(n,k);
```

参数说明： v 是返回的组合数结果。n 和 k 是输入参数，相当于组合数 C_n^k 中的 n 和 k 。

这里给出一个调用函数 beta 的例子，自变量取值范围是从 0~6。计算和绘图程序如下：

```
z=linspace(0,6,401);      % z 变量离散采样
B1=beta(z,4);            % 计算 Beta 函数
B2=beta(z,7);            % 计算 Beta 函数
B1L=betaln(z,4);          % 计算 Beta 函数的对数
B2L=betaln(z,7);          % 计算 Beta 函数的对数
subplot(121);plot(z,B1,'k');hold on;plot(z,B2,'k:');axis square;          % 绘图
xlabel('{\itB}({\itz},{\itw})','Fontname','Times New roman','Fontsize',12);   %
X 轴标注
L1=legend('{\itB}({\itz},4)','{\itB}({\itz},7)',1);
subplot(122);plot(z,B1L,'k');hold on;plot(z,B2L,'k:');axis square;         % 绘图
xlabel('ln[{\itB}({\itz},{\itw})]','Fontname','Times New roman','Fontsize',12);
% Y 轴标注
L2=legend('ln[{\itB}({\itz},4)]','ln[{\itB}({\itz},7)]',1);
set([L1,L2],'Fontname','Times new roman');
```

运行上面的程序，所得图形如图 15.11 所示。从图中可以看出 Beta 函数在区间 $[0,1]$ 内变化较大，并很快趋于 0。这里只是对变量 z 进行离散采样，用户也可以对变量 w 进行离散采样并计算函数值。

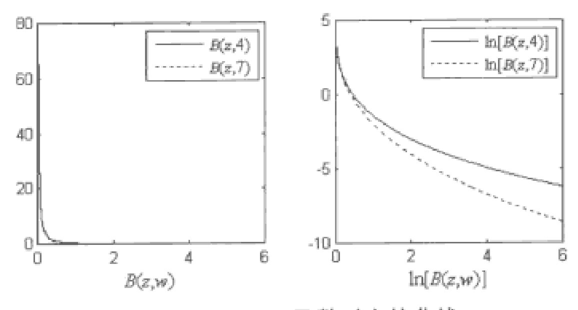


图 15.11　beta 函数对应的曲线

和 Gamma 函数一样，Beta 函数也存在不完全形式，即不完全 Beta 函数 $I_{1,x}(z,w)$ 和 $I_{2,x}(z,w)$，其数学定义为：

$$I_{1,x}=\frac{1}{B(z,w)}\int_0^x t^{z-1}(1-t)^{w-1}\,\mathrm{d}t \tag{15-15}$$

$$I_{2,x}=\frac{1}{B(z,w)}\int_x^1 t^{z-1}(1-t)^{w-1}\,\mathrm{d}t=1-I_{1,x}(z,w) \tag{15-16}$$

其中变量 x 定义于区间 $[0,1]$ 内。

MATLAB 提供了函数 betainc 来计算不完全 Beta 函数，其调用格式为：

```
I = betainc(x,z,w);
I = betainc(x,z,w,tail);
I = betainc(x,z,tail);
```

参数说明： I 是返回的不完全 Beta 函数值。x 对应于不完全 Beta 函数的自变量 x。z 和 w 是 Beta 函数的参数。参数 tail 用于指定不完全函数 $I_{1,x}(z,w)$ 和 $I_{2,x}(z,w)$，当 tail 取参数 lower 时计算 $I_{1,x}(z,w)$，当取参数 upper 时计算 $I_{2,x}(z,w)$，lower 为 tail 的默认值。

下面调用 betainc 函数来计算不完全 Beta 函数的数值，计算与绘图程序如下：

```
x=linspace(0,1,401);        % x 变量离散采样
I1=betainc(x,4,1,'upper');  % 计算不完全 Beta 函数
I2=betainc(x,4,2,'upper');  % 计算不完全 Beta 函数
I3=betainc(x,4,3);          % 计算不完全 Beta 函数
I4=betainc(x,4,4);          % 计算不完全 Beta 函数
plot(x,I1,'r');              % 绘图
hold on;
plot(x,I2,'g');              % 绘图
plot(x,I3,'b');              % 绘图
plot(x,I4,'k');              % 绘图
axis square;               % 设置坐标轴为方形
title('{\itI}({\itx},{\itz},{\itw})','Fontname','Times New
roman','Fontsize',14);       % 图题标注
xlabel('{\itx}','Fontname','Times New roman','Fontsize',14);                % X
轴标注
```

上述程序计算后得到如图 15.12 所示，不完全 Beta 函数的函数值介于区间 $[0,1]$ 之间。

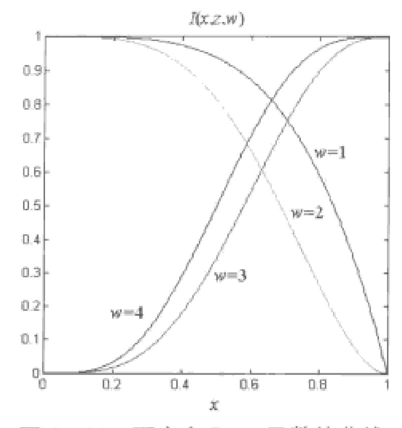


图 15.12 不完全 Beta 函数的曲线

15.5 其他特殊数学函数

本节介绍一些特殊函数，如雅可比椭圆函数。

雅可比椭圆函数的数学定义为：

$$u=\int_0^{\phi}\frac{\mathrm{d}\theta}{\sqrt{1-m\sin^2\theta}} \tag{15-17}$$

$$sn(u)=\sin\phi\text{，}\ cn(u)=\cos\phi\text{，}\ dn(u)=\sqrt{1-m\sin^2\phi}\text{，}\ am(u)=\phi \tag{15-18}$$

这里参数 m 要求是介于 0~1 之间的小数。

MATLAB 提供函数 ellipj 来计算雅可比椭圆函数，该函数的调用格式为：

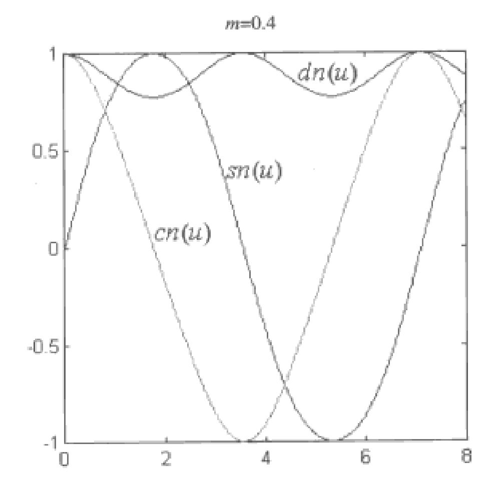
```
[sn,cn,dn] = ellipj(u,m);
[sn,cn,dn] = ellipj(u,m,tol);
```

参数说明：参数 sn，cn 和 dn 分别对应于 $sn(u)$，$cn(u)$ 和 $dn(u)$。u 是连续变量 u 的离散采样值。参数 m 用于指定 m 的数值。参数 tol 用于指定计算的精度，其默认值是 eps。

下面计算参数 m=0.4 和 m=0.8 时的雅可比椭圆函数曲线，实现程序如下：

```
u=linspace(0,8,401);                % 对 u 离散取样
[sn1,cn1,dn1] = ellipj(u,0.4);      % 计算 m=0.4 时的雅可比椭圆函数值
[sn2,cn2,dn2] = ellipj(u,0.8);      % 计算 m=0.8 时的雅可比椭圆函数值
subplot(121);
plot(u,sn1,'r');hold on;plot(u,cn1,'g');plot(u,dn1,'b');   % 绘图
axis square;                       % 设置坐标轴为方形
title(['m=',num2str(0.4)],'Fontsize',14,'Fontname','Times New Roman');
subplot(122);
plot(u,sn2,'r');hold on;plot(u,cn2,'g');plot(u,dn2,'b');   % 绘图
axis square;                       % 设置坐标轴为方形
title(['m=',num2str(0.8)],'Fontsize',14,'Fontname','Times New Roman');
```

程序执行结果如图 15.13 所示，函数 $cn(u)$ 和 $sn(u)$ 分布在-1 和 1 之间，而函数 $dn(u)$ 取值为正数。



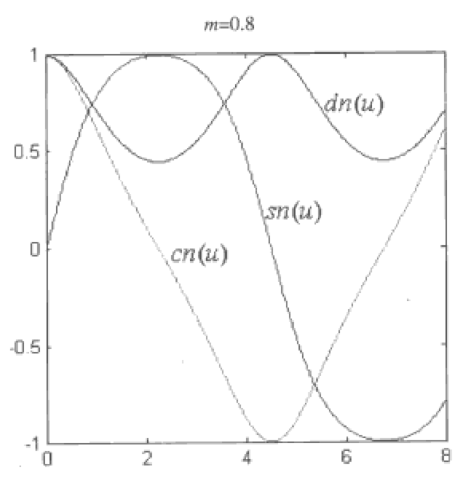


图 15.13 雅可比椭圆函数曲线

完整椭圆积分函数的数学定义为：

$$K(m)=\int_0^1 \frac{\mathrm{d}t}{\sqrt{\left(1-t^2\right)\left(1-mt^2\right)}}=\int_0^{\pi/2} \frac{\mathrm{d}\theta}{\sqrt{\left(1-m\sin^2\theta\right)}} \tag{15-19}$$

$$E(m)=\int_0^1 \frac{\sqrt{1-mt^2}\,\mathrm{d}t}{\sqrt{\left(1-t^2\right)}}=\int_0^{\pi/2} \sqrt{\left(1-m\sin^2\theta\right)}\mathrm{d}\theta \tag{15-20}$$

其中 $K(m)$ 和 $E(m)$ 分别是第一类和第二类椭圆函数，参数 m 为区间 $[0,1]$ 上的任意实数。

MATLAB 提供函数 ellipke 来计算椭圆函数，其调用格式为：

```
K = ellipke(m);
[K,E] = ellipke(m);
[K,E] = ellipke(m,tol);
```

参数说明： K 是第一类椭圆函数的函数值。E 是第二类椭圆函数的函数值。m 是椭圆函数的输入参数，其可以是一个数或者向量。tol 是精度控制量，其默认值为 eps。

下面利用函数 ellipke 来计算椭圆函数的曲线，实现程序如下：

```
m=linspace(0,1,401);      % 对 m 离散取样
[K,E] = ellipke(m);        % 计算椭圆函数值
plot(m,K,'r');hold on;plot(m,E,'k');    % 绘图
xlabel('\itm','Fontsize',14,'Fontname','Times new roman');  %标注
```

所得图形如图 15.14 所示，$K(m)$ 和 $E(m)$ 分别为单调递增和单调递减函数。

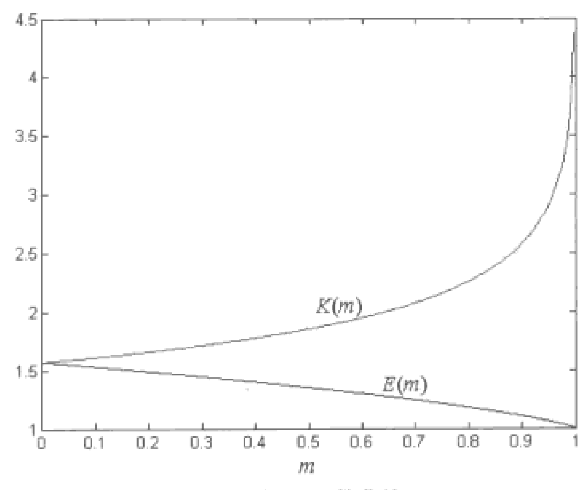


图 15.14 椭圆函数曲线图

误差函数 $\mathrm{erf}(x)$ 的数学定义为：

$$\mathrm{erf}(x)=\frac{2}{\sqrt{\pi}}\int_0^x \mathrm{e}^{-t^2}\mathrm{d}t \tag{15-21}$$

在 MATLAB 中提供了函数 erf 来计算误差函数，其调用格式为：

```
y = erf(x);
```

参数说明： y 是误差函数的数值。x 是误差函数自变量的离散采样值。

调用下面的语句可以计算出误差函数的曲线：

```
x=linspace(0,3,601);  % 对自变量进行离散采样
plot(x,erf(x));        % 计算函数值并绘图
```

所得误差函数的曲线如图 15.15 所示。从图 15.15 可以看出，当 $x>2$ 时误差函数几乎等于 1。

$\psi(x)$ 函数的数学定义为：

$$\psi(x)=\frac{\mathrm{d}\log\left[\Gamma(x)\right]}{\mathrm{d}x}=\frac{\mathrm{d}\Gamma(x)/\mathrm{d}t}{\Gamma(x)} \tag{15-22}$$

可见 $\psi(x)$ 就是 Gamma 函数自然对数的导数。当 $x=1$ 时，$-\psi(1)$ 等于欧拉常数（Euler's constant）。在 MATLAB 中提供了函数 psi 来计算 $\psi(x)$ 函数，其调用格式为：

```
y = psi(x);
y = psi(k,x);
```

参数说明： y 是输出的函数值。x 是自变量。k 用于指定派生的函数，其值为非负整数，默认值为 0。

调用下面的语句可以计算出 $\psi(x)$ 函数在区间 $[0.5,8]$ 内的曲线：

```
x = linspace(0.5,8,501);    % 数据离散取样
plot(x,psi(x));             % 计算并绘图
```

所得曲线如图 15.16 所示，该函数曲线在小于 1 时增加很快，最后增加速度变慢。

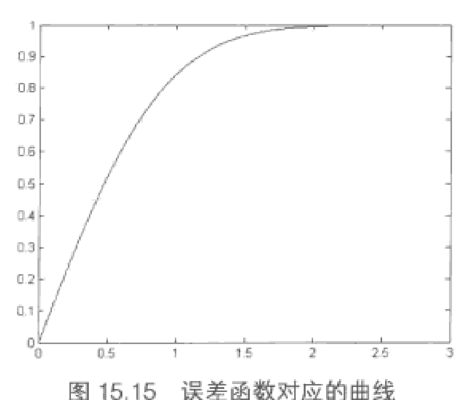


图 15.15　误差函数对应的曲线

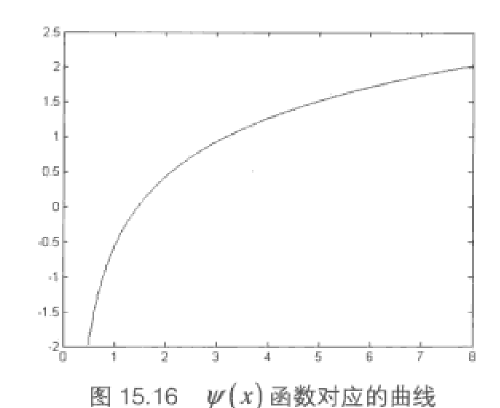


图 15.16　$\psi(x)$ 函数对应的曲线

一般而言，特殊函数可以展开为多项式的形式，如：

$$f(x)=\sum_{n=0}^{\infty}c_n x^n \tag{15-23}$$

如果次数高的幂函数项可以忽略，就可以仅计算有限项来获得函数值近似值。

这里以 $\sin x$ 为例取有限项进行近似计算，利用泰勒展开可知 $\sin x$ 可以表示为无穷级数之和。

$$\sin x=\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n+1}}{(2n+1)!} \tag{15-24}$$

这里计算到前 5 项的累加和并与理论解进行比较，可以根据式（15-24）进行计算。相关的

MATLAB 程序如下：

```
x=linspace(0,4,401);        % 对自变量离散取样
S1=0;                       % 初始化累加变量的数值
for n=0:4;
    S1=S1+(-1)^n/factorial(2*n+1)*x.^(2*n+1);    % 累加当前项
    if n>1.5;
        subplot(1,3,n-1);          % 打开一个空的坐标轴
        plot(x,S1,'r');hold on;    % 绘制近似解曲线
        plot(x,sin(x),'k:');       % 绘制理论解
        axis square;               % 设置坐标轴为方形
    end
end
```

输出结果如图 15.17 所示。

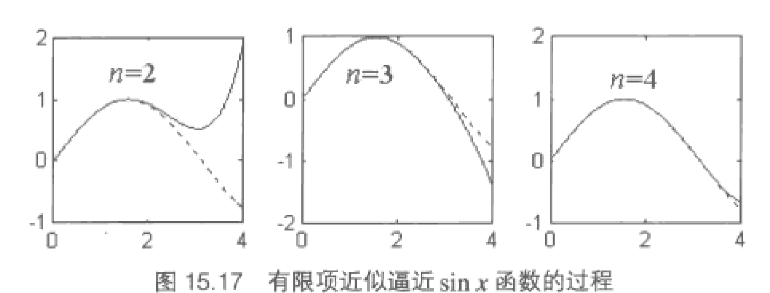


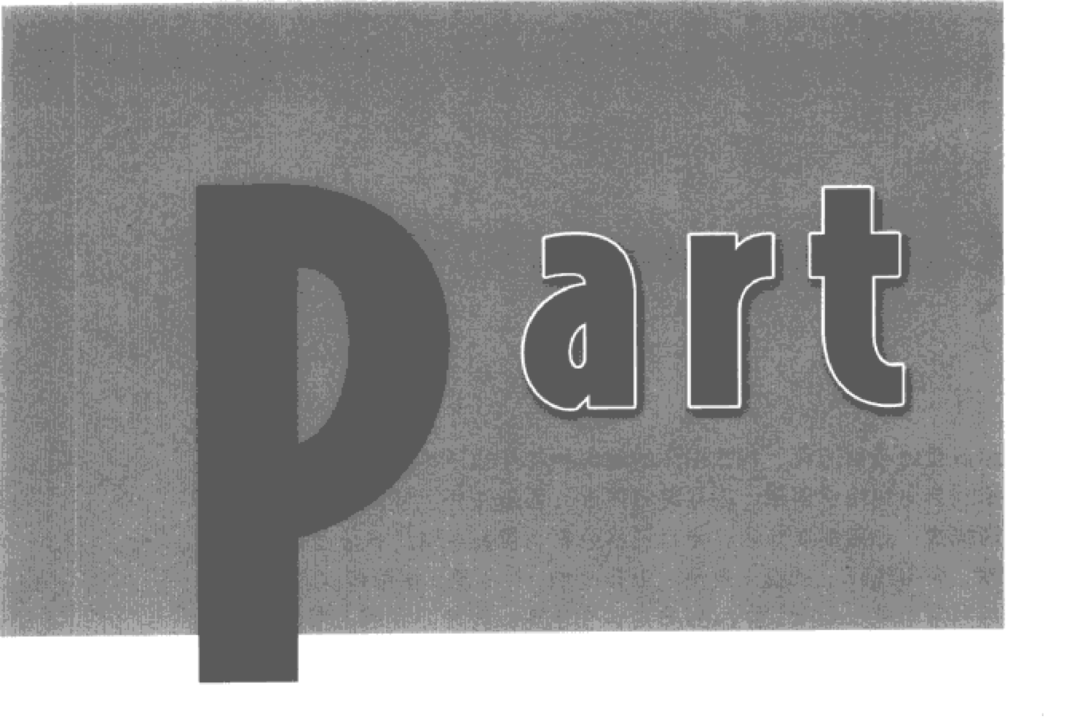
图 15.17　有限项近似逼近 $\sin x$ 函数的过程

从图 15.17 可以看出，随着选取项数的增加逐渐逼近于理论值。在取前 5 项时近似值和理论值之间的误差已经非常小。

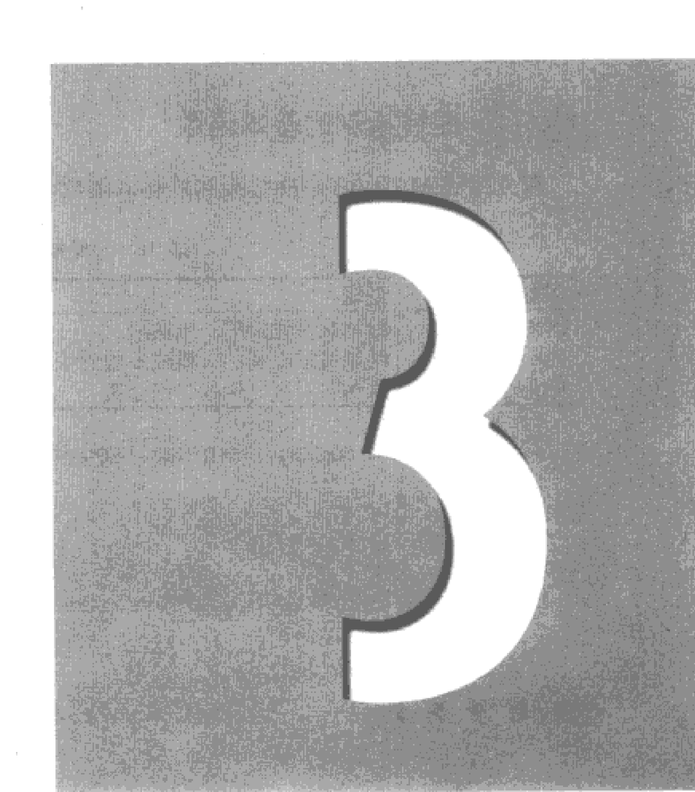
15.6　小结

本章主要介绍了特殊函数的计算。在 MATLAB 中提供了大多数常见特殊函数的数值计算程序，利用它们可以方便计算特殊函数。首先介绍了 5 种贝塞尔函数的计算及相关函数，如艾里函数。厄尔米特多项式函数是一类正交的多项式，而厄尔米特-高斯函数又是傅里叶变换（或者分数傅里叶变换）的本征函数，在 MATLAB 中调用 maple 函数可以计算该多项式的数值。此外本章还介绍了阶乘运算的 3 种计算方法，而 Gamma 函数可以看成一种广义阶乘，利用函数 gamma 可以得到另外一种计算阶乘的方法。最后介绍了 Beta 函数和其不完全形式的计算，以及雅可比椭圆函数、椭圆函数和误差函数等的计算方法。

◆　◆　◆



第3部分 数据可视化仿真



第 16 章 二维数据可视化

本章包括

- ◆ **基本命令** 介绍曲线的绘制函数、特殊图形的绘制和符号绘图等知识。
- ◆ **图形编辑** 介绍利用句柄编辑图形、鼠标编辑图形、图形注释、字体设定等。
- ◆ **自定义特殊图形样式** 介绍刻度标注、坐标网格绘制、箭头绘制、多值函数作图等。
- ◆ **基本图形的绘制** 介绍线段、弧线、矩形、正 N 边形、N 角星、圆管等。
- ◆ **多图布局** 介绍利用函数 subplot 和 axes 布局多个子图，同时介绍图上图的绘制。
- ◆ **图像处理函数** 介绍基本图像处理函数的用法。
- ◆ **动画** 介绍动画的绘制方法和相关函数，并给出两个例子。
- ◆ **图形的保存** 介绍利用函数 saveas 和 print 保存图形的方法。

数据的可视化是对结果的直观表示，在实际工作和科学研究中具有重要作用。而曲线表现数据变化规律是最直观的。本章主要介绍二维图形的绘制，通过合适的编辑可以得到我们所期望的图形样式。数值计算和可视化的结合是 MATLAB 的一个特色，MATLAB 提供了一系列相关的函数来完成绘图任务，其图形的可编辑性很强，用户可以对图形中的各部分按自己的需要样式进行编辑。用户可以结合工具条和鼠标对图形进行手动编辑，也可以用相关语句来等效地调整图形，两种方式各有其优点。本章将详细介绍不同二维图形的绘制和编辑方法。

16.1 基本命令

MATLAB 提供了具有不同功能的显示数据的曲线绘图函数，本节主要介绍这些函数的使用方法。

16.1.1 曲线绘制的基本函数

首先给出 MATLAB 中曲线绘制的基本函数，如表 16.1 所示。

表 16.1 绘制曲线的基本函数

函数名	说明	函数名	说明
plot	绘制线性比例的二维曲线	semilogx	绘制 X 轴为对数比例的曲线
line	绘制线性比例的二维或三维曲线	semilogy	绘制 Y 轴为对数比例的曲线
loglog	绘制双对数比例的二维曲线	plotyy	绘制双 Y 轴曲线

其中 plot，line，loglog，semilogx，semilogy 等函数的用法大体相同，可以统一描述为：

```
functionname(Ydata);
functionname(Xdata,Ydata);
functionname(Xdata,Ydata,LineSpec);
functionname(Xdata,Ydata, 'PropertyName1','PropertyValue1',...);
```

参数说明：functionname 表示上面提到的 5 个函数。Xdata 是 X 轴的数据，其默认值为 1:length(Ydata)。Ydata 是 Y 轴数据。LineSpec 用于指定曲线的线型、标记符号以及颜色等三项属性，它们的具体取值如表 16.2 所示。PropertyName1 表示曲线的属性名称，PropertyValue1 表示相应属性的取值。关于属性名称和取值的具体内容将在 16.2 节中详细说明，其中属性及其取值成对出现，并且可以有多对属性设定。

表 16.2 颜色、标记符号和线型控制

颜色		标记符号		线型	
控制符	说明	控制符	说明	控制符	说明
b	蓝色	.	实心圆	-	实线
g	绿色	o	空心圆	--	双画线
r	红色	x	斜十字	:	虚线
c	青色	+	十字符	-.	点画线
m	品红色	*	星号		
y	黄色	s	正方形		
k	黑色	d	菱形		
w	白色	v	下三角形(▽)		
		^	上三角形(△)		
		<	左三角形(◁)		
		>	右三角形(▷)		
		p	五角星(☆)		
		h	六角星(✡)		



在 LineSpec 中设定曲线的线型、标记符号以及颜色三项属性时，控制符的顺序不受限制。如'k-<'、'-<k'、'<k-'的效果是一样的。用户可以通过这些控制符来区分不同的曲线。

下面举例说明这几个函数的使用，个别函数的特殊用法同时给出。

16.1.1.1 plot 函数绘图

例 16-1：调用 plot 函数绘图的 3 种方式。

```
x=linspace(-2,2,21);          % 数据离散采样
subplot(131);
plot(x,sin(2*x),'k-<');          % [方式 1]绘图并设定线型、颜色以及标记符号等信息
axis square;                % 设定坐标轴为方形
subplot(132);
plot(x,cos(2*x),'k-*',2*x,x.*sin(x),'rs--');    % [方式 2]绘图并设定线型、颜色以及标记
符号等信息
axis square;                % 设定坐标轴为方形
subplot(133);
plot(1:10,rand(3,10));          % [方式 3]同时绘制多组数据
axis square;                % 设定坐标轴为方形
```

这里先得到 x 的离散采样数值，然后计算函数值并利用函数 plot 进行绘图。这里给出不同线型的结果。

在 MATLAB 中运行上述程序，所得图形如图 16.1 所示，不同线型可以给出不同的视觉效果，

用户可以根据自己的喜好在实际应用中选择。

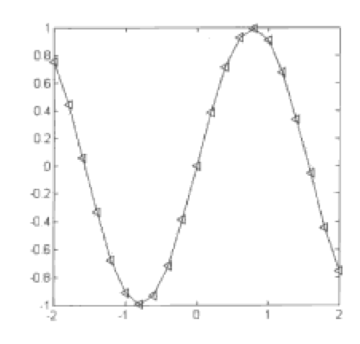
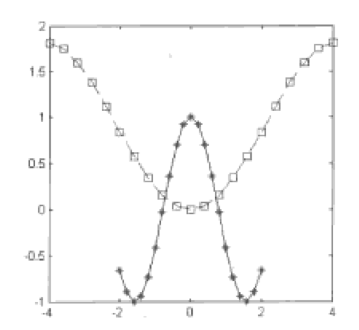
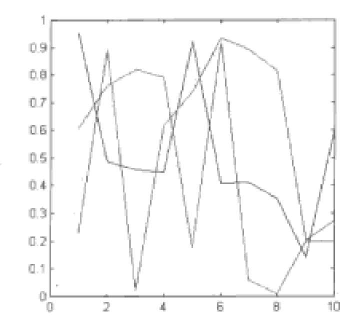




图 16.1 利用 plot 函数绘制曲线的例子



[方式 1]可以用来根据一组 X 轴数据和 Y 轴数据来绘制一条曲线，同时可以指定线型、颜色等信息；[方式 2]可以用来同时绘制多组曲线（而不必利用 hold on 语句），其中 X 轴数据范围可以不一样，需要注意的是用户只能对每条曲线的线型、颜色、标记符号进行不同的设定，而不能对其他属性进行相异的设置，比如这样一条语句“plot(x,cos(2*x),'k-*',2*x,x.*sin(x),'rs--','linewidth',2);”将每条曲线的线宽都设定为 2，而语句“plot(x,cos(2*x),'k-*','linewidth',3,2*x,x.*sin(x),'rs--');”是非法的；[方式 3]可以绘制 X 轴数据相同而 Y 轴数据不同的数据，如果用户指定曲线的属性，MATLAB 将自动地将不同颜色着色到相应数据上，着色顺序按表 16.2 中第一列自上而下进行，若超出 8 组数据，MATLAB 将自行添加不同的颜色。在[方式 3]中用户如果设定颜色，所有曲线将都变为相同颜色，如执行语句“plot(1:10,rand(3,10),'k');”后的效果是所有曲线都变为黑色。设定其他属性时也是针对所有曲线有效。

16.1.1.2 函数绘制曲线图

例 16-2：利用 line 函数绘制曲线。

```
t=linspace(0,pi*2,401);                 % 生成等间距采样点
subplot(131);
line(t,t.*sin(t*2),'LineStyle','-','Color','k');   % 绘制二维曲线
axis square;                             % 设定坐标轴为方形
xlim([min(t),max(t)]);                       % 设置 x 轴显示的数据范围
subplot(132);
line(t.*sin(4*t),t.*cos(4*t),t,'LineStyle','-','Color','k');  % 绘制三维曲线
axis square                              % 设定坐标轴为方形
subplot(133);
line(t.*sin(4*t),t.*cos(4*t),t,'LineStyle','-','Color','k');  % 绘制三维曲线
view(3)                                  % 三维视图
axis square                               % 设定坐标轴为方形
```

函数 line 和前面介绍的 plot 函数用法相似，读者可以进行对比来了解。上述程序运行后得到如图 16.2 所示的图形。

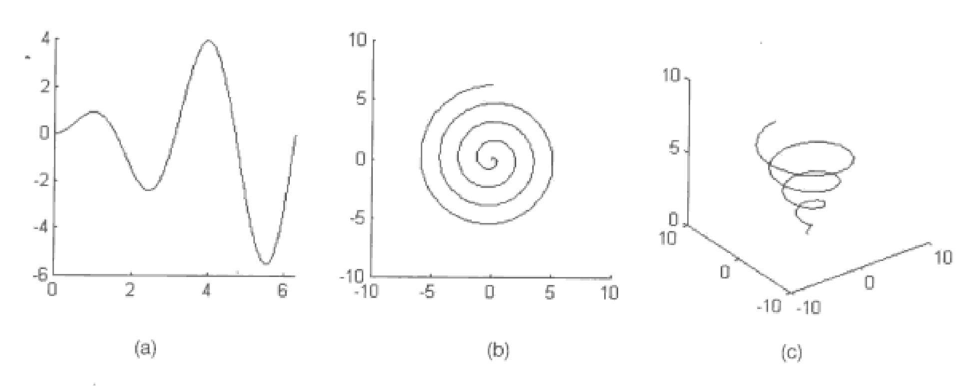


图 16.2 利用 line 函数绘制的曲线



对于二维曲线的绘制，如图 16.2(a)所示，函数 line 和函数 plot 用法相似，不同的是线型、颜色和标记符号需要分别使用 “LineStyle”、“Color” 和 “Marker” 来设定，而在 plot 函数使用中可以利用一个字符串来方便地设定。与函数 plot 不同的是函数 line 还可以用来画三维曲线，但函数 line 直接画出来的结果是三维曲线在 XY 平面上的投影，如图 16.2(b)所示，需要利用 view(3)语句来转化为三维视图。

函数 loglog，semilogx 和 semilogy 的用法和 plot 函数相同，这里不再赘述。

16.1.1.3 绘制双 Y 轴曲线

函数 plotyy 用来绘制双 Y 轴曲线，一般用于两组数据范围相差较大的情况。

例 16–3：绘制双 Y 轴曲线图。

```
x=linspace(0,2,201);                                        % 生成等间距的采样数据
subplot(131);
[Ax1,h1,h2]=plotyy(x,sin(x*2),x,sinh(exp(x)));              % [方式 1]
xlabel('(a)','Fontsize',14,'fontname','Times New Roman');   % X 轴标注
subplot(132);
[Ax2,h1,h2]=plotyy(x,abs(sin(x)),x,sinh(x),@semilogy);      % [方式 2]
xlabel('(b)','Fontsize',14,'fontname','Times New Roman');   % X 轴标注
box on;
subplot(133);
[Ax3,h1,h2]=plotyy(10*[x*10+1],abs(sin(x)),10*[x*10+1],sinh(x),
'loglog',@semilogx);  % [方式 3]
xlim(Ax3(1),[min(10*[x*10+1]),max(10*[x*10+1])]);        % 设置 X 轴范围
xlim(Ax3(2),[min(10*[x*10+1]),max(10*[x*10+1])]);        % 设置 X 轴范围
axis([Ax1,Ax2,Ax3],'square');                             % 设置坐标轴为方形
xlabel('(c)','Fontsize',14,'fontname','Times New Roman');  % X 轴标注
set([Ax1,Ax2,Ax3],'Fontsize',12);                           % 设置坐标轴字体
```

函数 plotyy 可以输入两组数据并给出两个 Y 轴不同刻度的图形样式，同时给出坐标轴标注的示例。上述程序执行后得到如图 16.3 所示的图形。

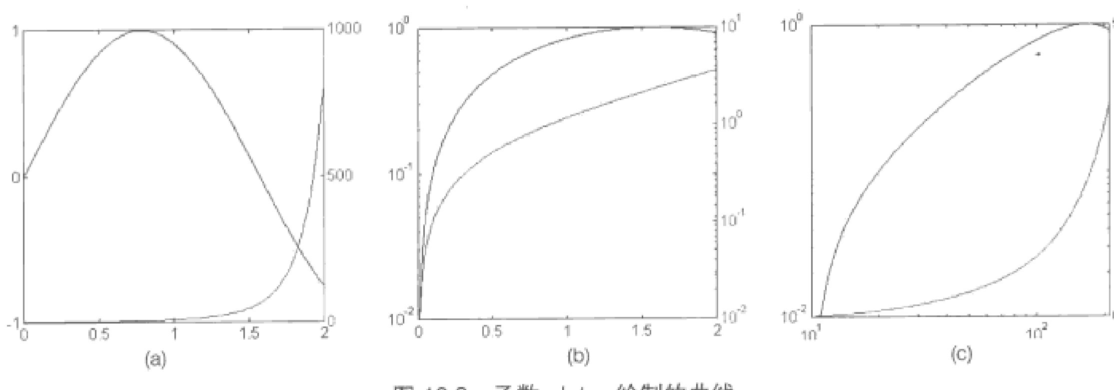


图 16.3 函数 plotyy 绘制的曲线

函数 plotyy 的调用格式比较简单，只有上面例子中的 3 种形式。[方式 1]绘制线性比例的双 Y 轴曲线；[方式 2]用于指定两条曲线使用对数比例绘图方式（可供选择的绘图函数还有 plot，semilogx，semilogy，loglog 和 stem 等，其中函数 plot 是默认值）；[方式 3]可以使用不同的绘图函数完成两条曲线的绘制。这里需要指出的是参数双 Y 轴曲线实际上是由两个坐标轴组成的，不同坐标轴的 Y 轴使用不同颜色加以区别，左侧 Y 轴用蓝色标记，右侧 Y 轴用绿色标记，某一颜色对应于相同颜色的 Y 轴刻度。设置坐标轴属性时需要对两个坐标轴同时设定。如语句：

```
axis([Ax1,Ax2,Ax3],'square');
xlim(Ax3(1),[min(10*[x*10+1]),max(10*[x*10+1])]);
xlim(Ax3(2),[min(10*[x*10+1]),max(10*[x*10+1])]);
```

参数说明：其中 Ax1，Ax2，Ax3 是1×2 的向量，是对应于两个坐标轴的句柄。函数 xlim 需要分别对两个坐标轴的范围进行设定。若一起设定，即：

```
xlim(Ax3,[min(10*[x*10+1]),max(10*[x*10+1])]);
```

将会出现错误提示：

```
??? Error using ==> xlim at 30
Wrong number of arguments
Error in ==> plotyy_test at 19
xlim(Ax3,[min(10*[x*10+1]),max(10*[x*10+1])]); % 设置 X 轴范围
```

在 MATLAB 中没有提供函数绘制双 X 轴曲线，不过用户可以在下述网页下载到实现这个功能的程序：http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=317。

该函数的调用格式为：

```
[ax,h1,h2] = plotxx(x1,y1,x2,y2,xlabels,ylabels);
```

参数说明：ax 是坐标轴的句柄。h1 和 h2 是两条曲线的句柄。x1，y1，x2，y2 是绘图数据。Xlabels，ylabels 是 X 轴和 Y 轴标注内容对应的细胞结构。

例 16-4：调用 plotxx 函数。

```
D = linspace(-100,0,50);     % Y 轴数据
 S = linspace(34,32,50);     % 上侧 X 轴数据
 T = 10*exp(D/40);          % 下侧 X 轴数据
 xlabels{1} = 'Temperature (C)';  % 下侧 X 轴标注内容
```

```
xlabels{2} = 'Salinity';          % 上侧 X 轴标注内容
ylabels{1} = 'Depth1(m)';        % 左侧 Y 轴标注内容
ylabels{2} = 'Depth2(m)';        % 右侧 Y 轴标注内容
[ax,hT,hS] = plotxx(T,D,S,D,xlabels,ylabels);  % 绘制双 X 轴曲线
```

上述程序执行后所得图形如图 16.4 所示。

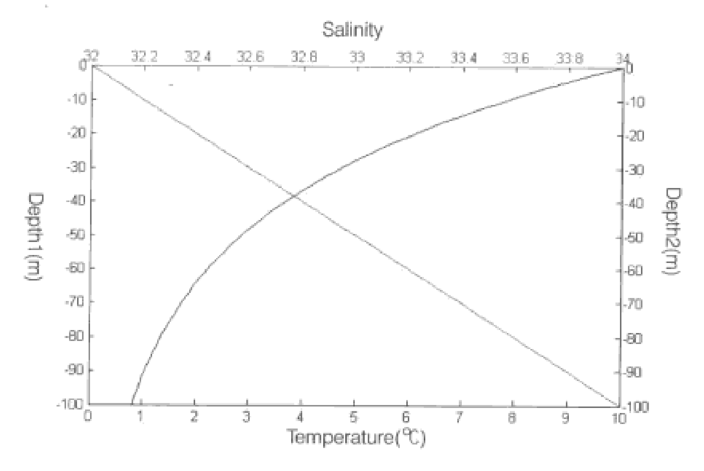


图 16.4 函数 plotxx 绘制的图形



其中下侧 X 轴曲线使用黑色标记，上侧 X 轴使用红色标记。

16.1.1.4 利用自编函数绘图

这里补充一个按曲线长度进行等间距取点的函数，即 samplep。这个函数是作者编写的一个实用小程序，该函数文件保存于光盘的\Ch16 文件夹中，其调用格式为：

```
[xn,yn]=samplep(x,y,N,ha);
```

参数说明： xn 和 yn 是返回的等间距数据。x 和 y 是数据曲线对应的数据。N 是返回的采样点数据。ha 是数据曲线所在的坐标轴句柄。

利用 samplep 函数绘制 $y=\exp\left(-x^2\right)/1000$，其中 $x\in[0,6]$，相应的程序如下：

```
x=linspace(0,6,200);  % 对自变量离散取样点坐标
y=exp(-x.^2)/1000;    % 计算相应的函数值
subplot(121);
plot(x,y,'k');         % 绘制曲线
axis square          % 设置当前坐标轴为方形
[xn,yn]=samplep(x,y,20,gca);
hold on;
plot(xn,yn,'ro');      % 用圆圈标记按曲线长度等间距取样点
xlabel('(a)','Fontsize',14,'Fontname','Times new roman');    % X 轴标注
subplot(122);
plot(x,y,'k');         % 绘制曲线
axis square          % 设置当前坐标轴为方形
hold on;
```

```
plot(x(1:10:end),y(1:10:end),'ro');                         % 用圆圈标记按 X 轴等间距取样点
xlabel('(b)','Fontsize',14,'Fontname','Times new roman');    % X 轴标注
```

这里首先利用函数 plot 画出曲线，然后利用函数 samplep 等间距取点，对所得点利用函数 plot 把相应的点以圆圈标记出来。上述程序执行后将得到如图 16.5 所示的图形。

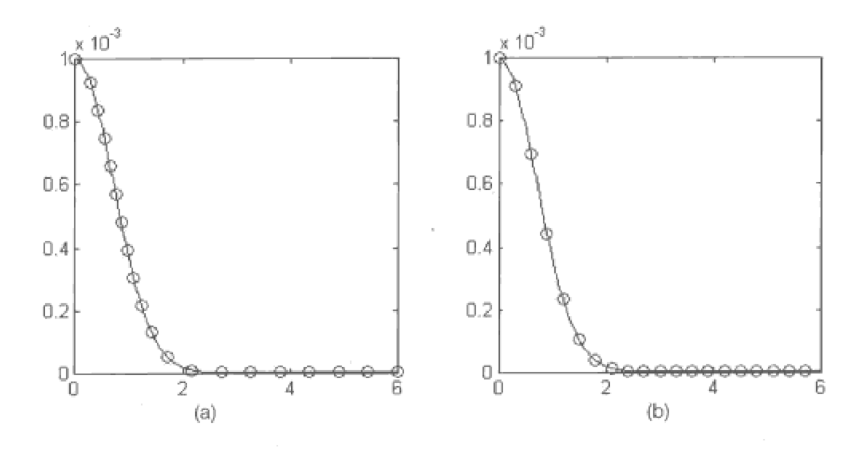


图 16.5　按曲线长度等间距采样的图形



为了对比效果，同时给出了利用 X 轴等间距采样的结果（如图 16.5(a)所示）。图 16.5(b)中在水平方向上圆圈密集，但是在竖直方向上稀疏，而图 16.5(a)在任何方向上都是按曲线长度间隔相等取点的。

16.1.2　特殊图形的函数

除了上面介绍的关于曲线绘制的函数外，MATLAB 还提供了绘制一些特殊图形的函数，具体函数及功能如表 16.3 所示。

表 16.3　绘制特殊图形的函数

函数名	功能	函数名	功能
bar, barh	垂直和水平直方图	pie	饼图
compass	罗盘图	polar	极坐标图
contour	等高线图	quiver	向量场图
errorbar	在曲线上添加误差范围	rose	极坐标累计图
feather	羽毛图	stairs	阶梯图
hist	频数统计直方图	stem	针状图
pareto	排列图（帕累托图）		

下面来介绍这些函数的使用方法。

16.1.2.1　直方图

函数 bar 和 barh 可以用来画垂直和水平直方图，函数 bar 的调用格式为：

```
bar(y);
```

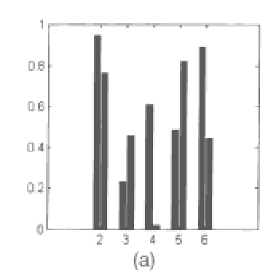
```
bar(x,y);
bar(x,y,width)
bar(...,linespec);
bar(...,'format');
```

参数说明： y 是待统计的数据，它是向量或者数组。x 是 x 轴的坐标，其默认值是 1:size(y,1)。x 的元素数目要求和 y 的行数相等。width 用于指定 bar 的宽度，当 width>1 时相邻的 bar 之间会存在重叠的情况，width 的默认值是 0.8。linspec 是一个单个字符，用于指定所有条形图的颜色，可选值为 r，g，b，y，m，c，k，w 之一，其他颜色需要通过句柄来设定，其默认时，每组条状图内用不同颜色区分各个条。参数 format 用于指定条状图的分布方式，可选择为 grouped（排列型条状图）和 stacked（堆型条状图），其中 grouped 为默认值。

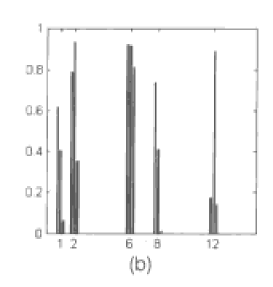
下面调用 bar 函数绘制条状图，计算和绘图程序如下：

```
x1=2:6;
x2=[1,2,6,8,12];
s(1)=subplot(141);
bar(x1,rand(5,2));     % 绘制条状图，宽度、颜色及排列方式都默认
xlabel('(a)','Fontsize',14,'Fontname','Times New Roman');  % X 轴标注
s(2)=subplot(142);
bar(x2,rand(5,3),0.4,'k');    % 绘制条状图，指定宽度和颜色
xlabel('(b)','Fontsize',14,'Fontname','Times New Roman');  % X 轴标注
s(3)=subplot(143);
bar(x1,rand(5,3),1.6);              % 绘制条状图，指定宽度大于 1
xlabel('(c)','Fontsize',14,'Fontname','Times New Roman');  % X 轴标注
s(4)=subplot(144);
bar(x2,rand(5,4),0.4,'stacked');    % 绘制条状图，指定堆型排列
axis(s,'square');                   % 设置所有坐标轴为方形
xlabel('(d)','Fontsize',14,'Fontname','Times New Roman');  % X 轴标注
```

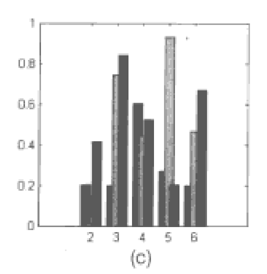
这里给出横坐标不同取点方式，以及不同排布方式的条状图样式。条的宽度可以根据需要进行设定。程序执行后所得图形如图 16.6 所示。



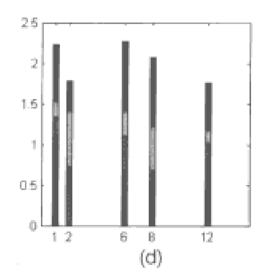
(a)



(b)



(c)



(d)

图 16.6 利用函数 bar 绘制的条状图



在图 16.6 中，图(b)的所有条形图都是黑色的，其他的为彩色条。图(c)中存在着交叠的情况。

函数 bar 是绘制竖直方向的条状图，而函数 barh 是绘制水平方向的条状图，函数 barh 和函数 bar 的用法相同，把前面调用函数 bar 的例子中“bar”改为“barh”，可以得到如图 16.7 所示的横向条状图。

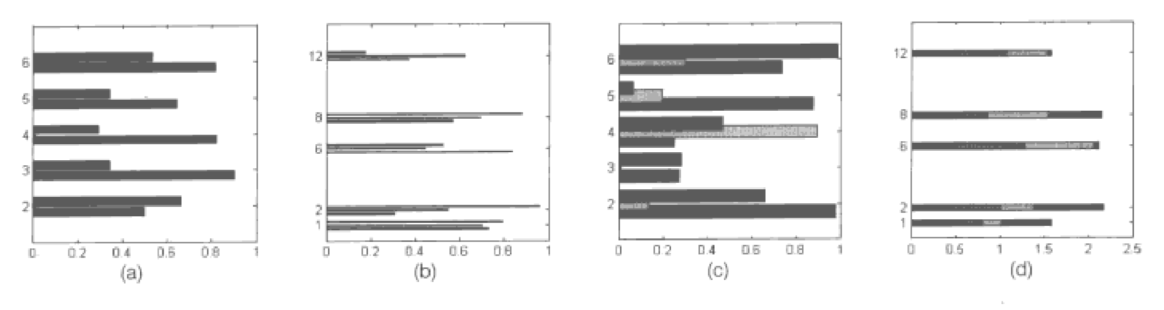


图 16.7　函数 barh 绘制的横向条状图

16.1.2.2　罗盘图

函数 compass 用来绘制罗盘图，即在极坐标系下用带箭头的线段来表示一个复数，其调用格式为：

```
compass(u,v);
compass(z);
```

参数说明： u 和 v 分别为复数数组的实部和虚部。z 是一个复数数组。

例 16–5： 调用函数 compass。

```
u=randn(2,3);              % 生成实部对应的数据
v=randn(2,3);              % 生成虚部对应的数据
z=exp(i*randn(1,6));         % 生成一个复数数组
subplot(121);
compass(u,v);              % 利用实部和虚部绘制罗盘图
xlabel('(a)','Fontsize',14,'Fontname','Times New Roman');   % X 轴标注
subplot(122);
compass(z);               % 根据复数绘制罗盘图
xlabel('(b)','Fontsize',14,'Fontname','Times New Roman');   % X 轴标注
```

此处以随机数作为罗盘图的输入数据，可以以实部和虚部作为函数 compass 的输入，也可以用复数组成的向量作为输入。所得图形如图 16.8 所示，每个箭头的长度和方向分别对应着输入复数的模值和相角。 (a) (b) (c)

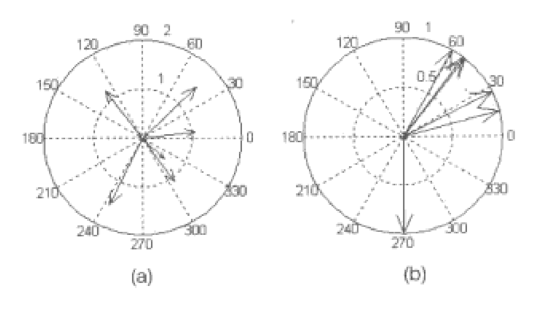


图 16.8　罗盘图

16.1.2.3　等高线图

函数 contour 用来绘制等高线图，其调用格式为：

```
contour(z);
contour(x,y,z)
contour(z,n);
contour(z,v);
```

参数说明：z 是等高线的高度数据。x 和 y 分别用于指定 X 轴和 Y 轴的坐标。n 表示等高线的数目。v 用来指定高度值处的等高线。

例 16–6：调用函数 contour 绘制等高线。

```
[x,y,z]=peaks(80);     % 生成坐标刻度数据和高度数据
s(1)=subplot(141);
contour(z);           % 根据高度数据绘制等高线图
xlabel('(a)','Fontsize',14,'Fontname','Times New Roman');  % X轴标注
s(2)=subplot(142);
contour(x,y,z);        % 根据刻度数据和高度数据绘制等高线图
xlabel('(b)','Fontsize',14,'Fontname','Times New Roman');  % X轴标注
s(3)=subplot(143);
contour(x,y,z,4);      % 指定等高线的数目
xlabel('(c)','Fontsize',14,'Fontname','Times New Roman');  % X轴标注
s(4)=subplot(144);
contour(x,y,z,linspace(min(z(:)),max(z(:)),12));  % 等间隔指定高度位置
xlabel('(d)','Fontsize',14,'Fontname','Times New Roman');  % X轴标注
axis(s,'square');        % 设置所有坐标轴为方形
```

这里利用函数 peaks 得到相应的数据 x，y 和 z。然后给出函数 contour 的不同调用格式下的等高线图。输出图形如图 16.9 所示，在图(a)中的坐标刻度是对应于矩阵 $\boldsymbol{Z}$ 的行数和列数，而图(b)、图(c)和图(d)的刻度是根据矩阵 $\boldsymbol{X}$ 和 $\boldsymbol{Y}$ 生成的。

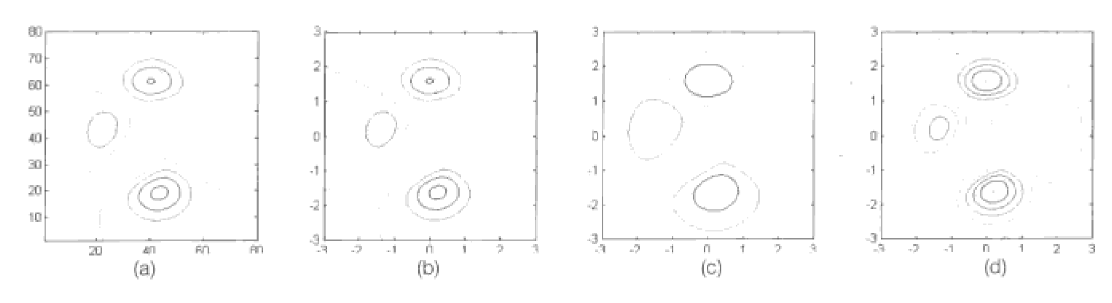


图 16.9　等高线图

16.1.2.4　在曲线上添加误差范围

函数 errorbar 可以实现在曲线上添加误差范围，其调用格式为：

```
errorbar(x,y,e);
errorbar(x,y,L,U);
```

参数说明：x 和 y 分别是 X 轴和 Y 轴的数据。e 表示误差量，y 的范围是[y(k)-e(k),y(k)+e(k)]。L 和 U 分别是上侧误差和下侧误差，此时 y 的范围是[y(k)-L(k),y(k)+U(k)]。

例 16–7：调用 errorbar 函数绘图。

```
x = 1:10;
y = sin(x);
```

```
e = std(y)*ones(size(x))/6;  % 误差范围
L=rand(1,10)/5;              % 下侧误差范围
U=rand(1,10)/5;              % 上侧误差范围
s(1)=subplot(121);
errorbar(x,y,e);              % 绘制带误差条的曲线
xlabel('(a)','Fontsize',14,'Fontname','Times New Roman');  % X 轴标注
s(2)=subplot(122);
errorbar(x,y,L,U);            % 绘制带误差条的曲线
xlabel('(b)','Fontsize',14,'Fontname','Times New Roman');  % X 轴标注
axis(s,'square');              % 设置所有坐标轴为方形
```

这里曲线是通过函数 sin(x)给出相应的数据而利用随机函数生成的误差。调用 errorbar 给出带误差条的曲线图。所得图形如图 16.10 所示，其中图(a)是误差量上下相等，而图(b)中曲线上下两侧误差不相等。

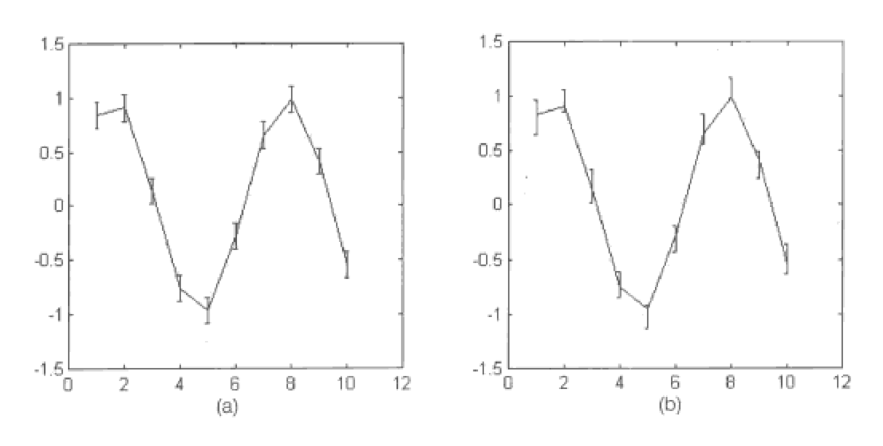


图 16.10　带误差条的曲线

16.1.2.5　羽毛图

函数 feather 可以绘制羽毛图，其调用格式为：

```
feather(u,v)
feather(z)
```

参数说明： u 和 v 分别是复数的实部和虚部。z 是一个复数数组。

例 16-8： 调用 feather 绘制羽毛图。

```
theta = (-90:10:90)*pi/180;   % 生成角度数据
r = 2*ones(size(theta));      % 生成半径数据
[u,v] = pol2cart(theta,r);    % 把极坐标转化为直角坐标
feather(u,v);                 % 绘制羽毛图
axis equal                    % 设置坐标轴刻度相等
```

所得图形如图 16.11 所示，因为图形看上去像羽毛，所以称之为羽毛图。

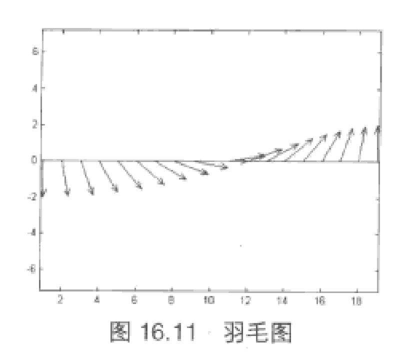


图 16.11 羽毛图

16.1.2.6 频数统计直方图

函数 hist 用来实现频数统计直方图的绘制，其调用格式为：

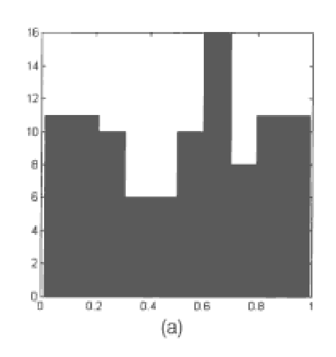
```
hist(y);
n = hist(y,m);
n = hist(y,x);
[n,x] = hist(...);
```

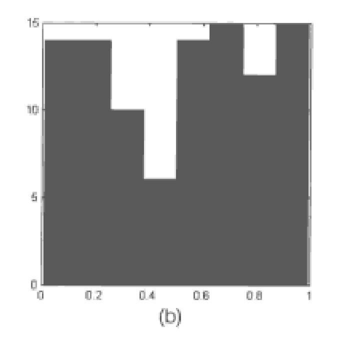
参数说明： y 是待统计的数据。m 是整数，用于指定统计区域的个数。x 是向量，指定每个统计区域的 X 轴刻度。n 表示每个区域内数值的个数。当无返回值时函数 hist 将绘制统计直方图。

例 16-9： 调用函数绘制统计直方图。

```
y=rand(100,1);      % 生成待统计的数据
[n,x]=hist(y)         % 返回统计频数 n 和区域中心位置 x
s(1)=subplot(131);
hist(y);              % 绘制统计直方图
xlabel('(a)','Fontsize',14,'Fontname','Times New Roman');  % X 轴标注
s(2)=subplot(132);
hist(y,8);            % 绘制统计直方图，并指定区域数目
xlabel('(b)','Fontsize',14,'Fontname','Times New Roman');  % X 轴标注
s(3)=subplot(133);
hist(y,0:.1:1);        % 绘制统计直方图，并指定每个区域的中心位置
xlabel('(c)','Fontsize',14,'Fontname','Times New Roman');  % X 轴标注
axis(s,'square');     % 设置所有坐标轴为方形
```

此处以随机数据作为输入来绘制其统计直方图，同时给出了 3 种不同样式的统计直方图，输出图形如图 16.12 所示，从图中可以看出不同数值处数据个数的统计结果。





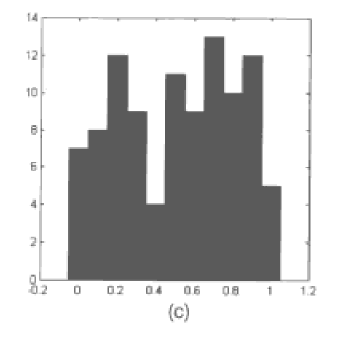


图 16.12 频数统计直方图

16.1.2.7 排列图

函数 pareto 可用来绘制排列图，其调用格式为：

```
pareto(y,names);
pareto(y,x);
```

参数说明： y 是条形图的高度值。x 是对应于 y 值的 X 轴坐标。names 是对应于 y 元素的 X 轴标注，其值为字符串。

例 16-10： 调用 pareto 函数。

```
y1=rand(1,6);                         % 生成第一组 Y 轴数据
y2=rand(1,6);                         % 生成第二组 Y 轴数据
x=2:7;                              % 生成 X 轴数据点坐标
names={'Ax','By','C0','D','E1','F2'};  % 生成对应的条状图的坐标
subplot(121);[H1,ax1]=pareto(y1,x);              % 绘制排列图
xlabel('(a)','Fontsize',14,'Fontname','Times New Roman');  % X 轴标注
subplot(122);
[H2,ax2]=pareto(y2,names);           % 绘制帕累托图
xlabel('(b)','Fontsize',14,'Fontname','Times New Roman');  % X 轴标注
set([H1(2),H2(2)],'Marker','*');           % 设置数据点标记符号为星形
```

这里以随机数据作为输入，对于 X 轴刻度同时给出两种不同方式的标注方法，所得图形如图 16.13 所示。

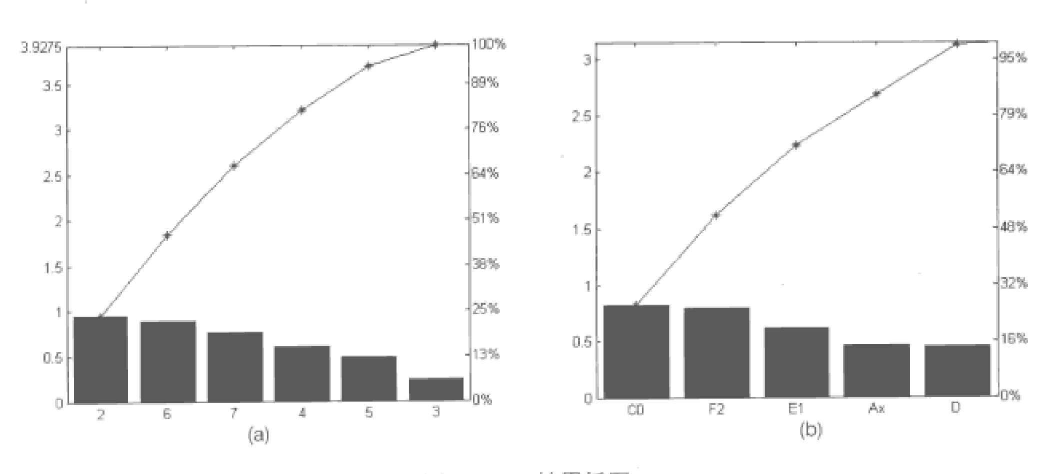


图 16.13 帕累托图



高度数据 y 要求是非负且不能是 NaN。高度数据按照由大到小顺序排列。Y 轴的坐标范围是[0,sum(y)]，右侧 Y 轴刻度对应于百分比。曲线表示高度数据的累积分布。帕累托图在统计问题中具有重要应用。

16.1.2.8 饼图

函数 pie 用于绘制饼图，其调用格式为：

```
pie(x)
```

```
pie(x,explode)
pie(...,labels)
```

参数说明： x 是欲分析数据对应的数组。Explode 与 x 的维数相同，其中的非零元素对应的切片就是分离的切片。labels 中的参数是增加的自定义标注内容，其与 x 的维数相同。

例 16–11： 调用函数 pie 绘制饼图。

```
x1=rand(1,5);  % 生成第一组 x 数据
x2=rand(2,3);  % 生成第二组 x 数据
x3=rand(1,4);  % 生成第三组 x 数据
labels={'Ax1','C0','E2','F0'};   % 生成对应的切片的名称
subplot(131);pie(x1);          % 绘制饼图
subplot(132);pie(x2,[0,1,0;0,0,0]);   % 绘制饼图
subplot(133);pie(x3,labels);        % 绘制饼图
```

上述程序计算所得图形如图 16.14 所示。

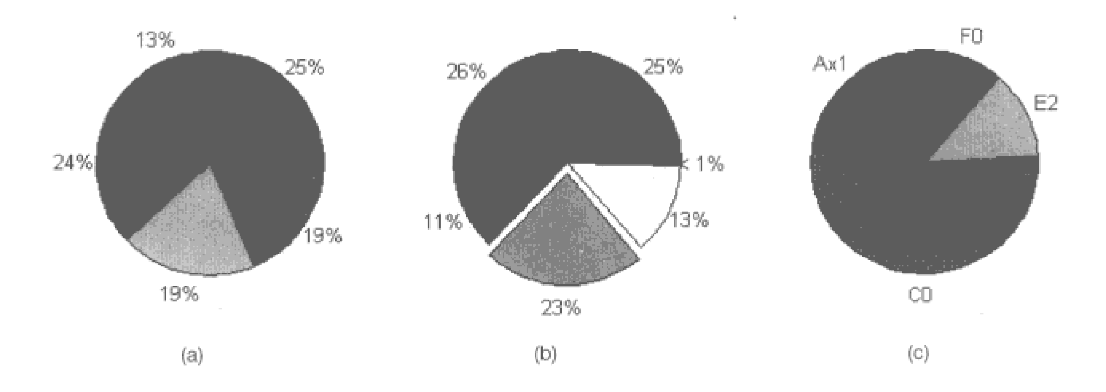


图 16.14　函数 pie 绘制的饼图



在图 16.14 中，图(c)自定义的标注代替了各切片的百分比。

16.1.2.9　极坐标图

函数 polar 用于绘制极坐标图，其调用格式为：

```
polar(theta, rho)
polar(theta,rho,s)
```

参数说明： theta 是角度数据。rho 是极径方向的数据。s 是用于指定线型、标记符号和颜色的字符串，可选值和函数 plot 相同。

例 16–12： 利用函数 polar 绘制极坐标。

```
theta=linspace(0,pi*2,101);    % 生成角度数据
rho=sin(theta+cos(theta));    % 生成极径数据
subplot(121);polar(theta,rho);  % 绘制极坐标图
xlabel('(a)','Fontsize',14,'Fontname','Times New Roman');
subplot(122);polar(theta,sin(theta*2+sin(4*theta)),'k-x');     % 绘制极坐标图
```

```
xlabel('(b)','Fontsize',14,'Fontname','Times New Roman');
```

所得图形如图 16.15 所示，两幅极坐标图中的曲线是封闭的，同时图(b)在曲线上增加了“乘号”标记。

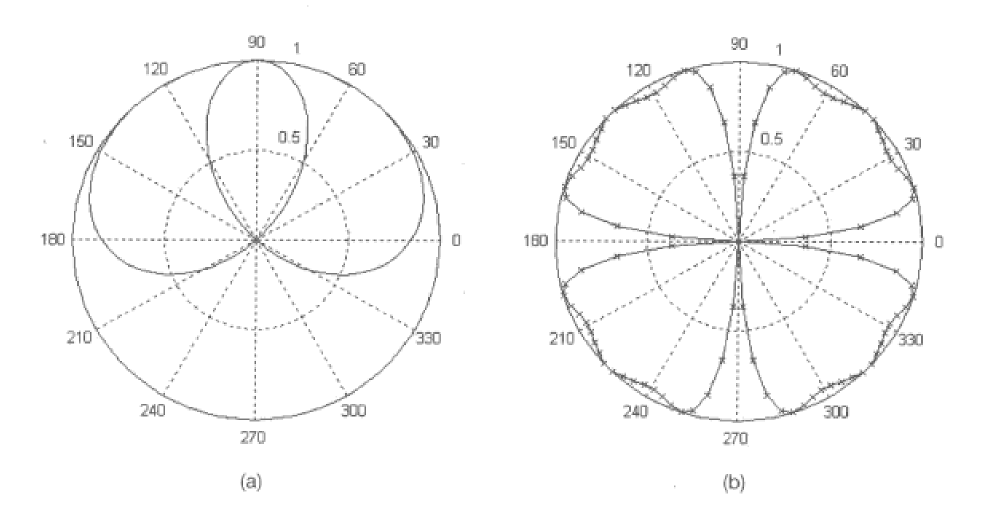


图 16.15 极坐标图

16.1.2.10 向量场图

函数 quiver 可以绘制向量场图，其调用格式为：

```
quiver(x,y,u,v);
quiver(u,v);
quiver(...,s);
quiver(...,linespec)
quiver(...,'filled')
```

参数说明： x 和 y 是指定箭头位置的坐标。u 和 v 分别是向量场水平和竖直分量的大小。s 表示缩放的比例，当 s=0 时向量为默认长度 1，此时 MATLAB 自动调整缩放比例以防止箭头发生重叠的情况。linespec 用于指定箭头的线型、颜色和标记符号等。filled 表示使用 linespec 中的标记符号填充箭头的位置。

下面调用函数 quiver 绘制向量场图。

```
[x,y] = meshgrid(-2:.2:2,-1:.15:1);    % 生成坐标网格
z = x .* exp(-x.^2 - y.^2);             % 计算二元函数的离散函数值
[px,py] = gradient(z,.2,.15);          % 计算梯度
subplot(221);quiver(x,y,px,py);axis image;    % 绘制向量场图
xlabel('(a)','Fontsize',14,'Fontname','Times New Roman');  % X 轴标注
subplot(222);quiver(px,py,1);axis image;     % 绘制向量场图
xlabel('(b)','Fontsize',14,'Fontname','Times New Roman');  % X 轴标注
subplot(223);quiver(px,py,2,'k-.');axis image;  % 绘制向量场图
xlabel('(c)','Fontsize',14,'Fontname','Times New Roman');  % X 轴标注
subplot(224);quiver(px,py,1,'x','filled');axis image;   % 绘制向量场图
xlabel('(d)','Fontsize',14,'Fontname','Times New Roman');  % X 轴标注
```

上面代码首先给出一个二元函数作为输入，随后计算其梯度值作为采样点上的矢量方向表示，同时给出 4 种不同输入参数情况下的矢量图。所得图形如图 16.16 所示，可见不同参数情况下坐标轴显示的大小有所不同，同时表示矢量箭头的大小也因参数而异。

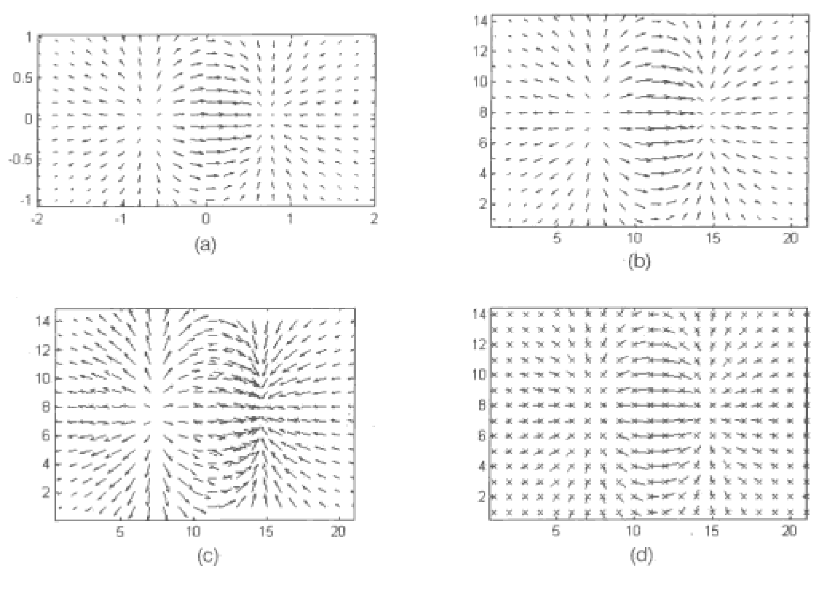


图 16.16 向量场图

16.1.2.11 坐标累积图

函数 rose 用于绘制极坐标累积图，其调用格式为：

```
rose(theta);
rose(theta,n);
rose(theta,x);
```

参数说明： theta 为角度数据。n 用于定义小扇形的个数。x 用于指定小扇形的位置，其维数由小扇形的个数决定。

下面调用函数 rose 绘制极坐标系下的累积图：

```
theta=randn(1,60);
subplot(131);rose(theta);      % 绘制极坐标累积图
xlabel('(a)','Fontsize',14,'Fontname','Times New Roman');  % X 轴标注
subplot(132);rose(theta,8);    % 绘制极坐标累积图
xlabel('(b)','Fontsize',14,'Fontname','Times New Roman');  % X 轴标注
subplot(133);rose(theta,[1,2,4]);  % 绘制极坐标累积图
xlabel('(c)','Fontsize',14,'Fontname','Times New Roman');  % X 轴标注
```

上述程序输出结果如图 16.17 所示，这里以正态分布的随机数作为输入，给出了 3 种不同输入参数时的累积图绘制例子。

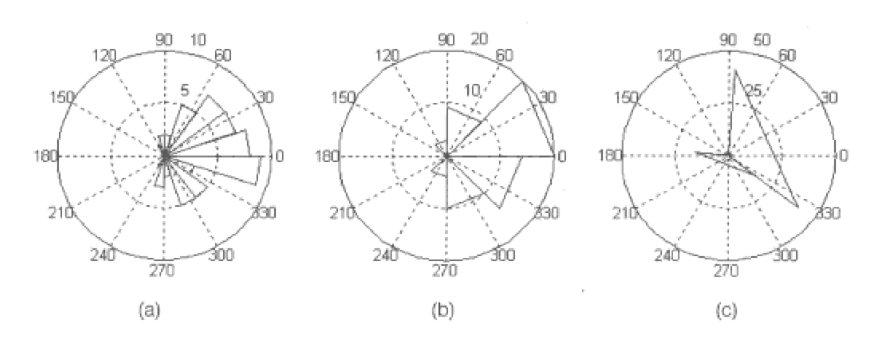


图 16.17　极坐标累积图

16.1.2.12　阶梯图

函数 stairs 可用于绘制阶梯图（以一定间隔沿上升或者下降方式显示数据），其调用格式为：

```
stairs(y);
stairs(x,y);
stairs(...,style);
```

参数说明： x 和 y 为横纵坐标的数据。style 来指定绘制阶梯图时的线型、颜色和标记符号等。

下面调用函数 stairs 来绘制阶梯图：

```
x=rand(1,6);                    % 生成横坐标数据
y=rand(1,6);                    % 生成纵坐标数据
s(1)=subplot(131);stairs(y);    % 绘制阶梯图
xlabel('(a)','Fontsize',14,'Fontname','Times New Roman');   % X 轴标注
s(2)=subplot(132);stairs(x,y);  % 绘制阶梯图
hold on;plot(x,y,'ro');
xlabel('(b)','Fontsize',14,'Fontname','Times New Roman');   % X 轴标注
s(3)=subplot(133);stairs(1:6,y,'k-*');   % 绘制阶梯图
xlabel('(c)','Fontsize',14,'Fontname','Times New Roman');   % X 轴标注
axis(s,'square');                         % 设置所有坐标轴为方形
```

在这个例子中，使用随机数作为 Y 轴输入，而 X 轴的数据使用默认、随机数以及指定的向量 3 种方式下绘制的阶梯图结果如图 16.18 所示，可以根据数据位置理解阶梯图绘制时输入数据的顺序。

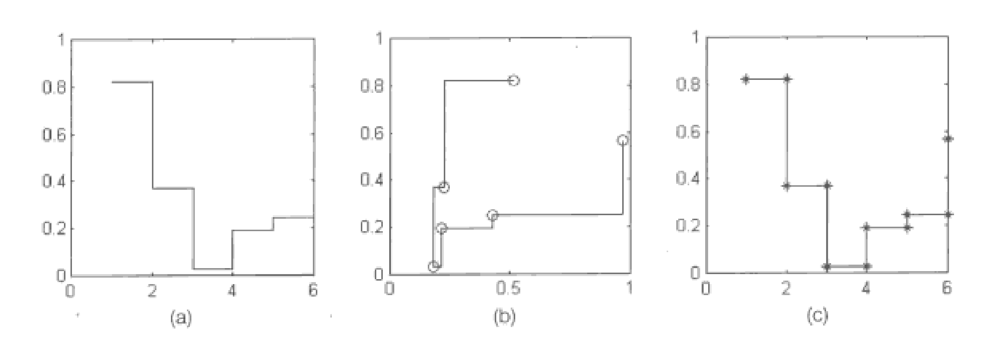


图 16.18　阶梯图

16.1.2.13 针状图

函数 stem 可以绘制针状图，其调用格式为：

```
stem(y);
stem(x,y)
stem(...,'linespec')
stem(...,'filled')
```

参数说明： x 和 y 分别是横、纵坐标的数据点，其中 x 的默认值为[1:length(y)]。linespec 用于指定线型、颜色和标记符号等。filled 指用标记符号填充针状图。

例 16–13： 调用函数 stem 绘制针状图。

```
y=rand(1,6);                              % 生成纵坐标数据
s(1)=subplot(131);stem(1:6,y);  % 绘制针状图
xlabel('(a)','Fontsize',14,'Fontname','Times New Roman');  % X 轴标注
s(2)=subplot(132);stem(1:2:11,y,'k:s');                % 绘制针状图
xlabel('(b)','Fontsize',14,'Fontname','Times New Roman');   % X 轴标注
s(3)=subplot(133);stem(linspace(0,1,6),y,'r*','fill');          % 绘制针状图
xlabel('(c)','Fontsize',14,'Fontname','Times New Roman');   % X 轴标注
axis(s,'square');                                  % 设置所有坐标轴为方形
```

这里以均匀分布的随机数作为输入，给出 3 种针状图绘制的例子，所得图形如图 16.19 所示，其中“针头”分别以“圆圈”、“方块”和“*”来显示。

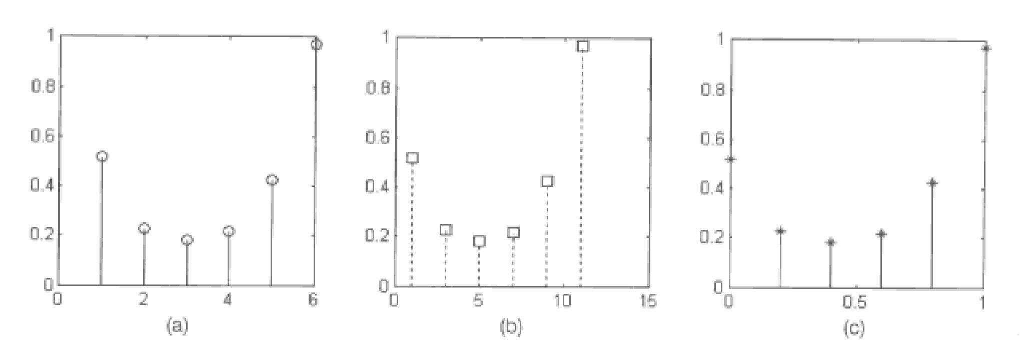


图 16.19 针状图

16.1.3 符号绘图

除了前一小节介绍利用离散数据绘图的函数外，MATLAB 还可以根据函数的表达式来绘制曲线。本小节介绍相关函数的用法。MATLAB 提供的二维符号绘图函数如表 16.4 所示。

表 16.4 二维符号绘图函数

函数名	说明	函数名	说明
fplot	函数绘图	ezcontourf	等高线
ezplot	符号绘图	ezpolar	极坐标图
ezcontour	等高线		

16.1.3.1 fplot 曲线图

函数 fplot 可以指定函数并绘制曲线，其调用格式为：

```
fplot(fun,lims);
fplot(fun,lims,tol);
fplot(fun,lims,n);
fplot(fun,lims,'linespec')
```

参数说明： fun 用来表达函数，其可以是常用数学函数，也可以是用户自定义的函数。lims 用于指定坐标轴的范围。tol 用于指定相对误差精度。n 用于指定绘图的最少点为 n+1，因此最大步长为(xmax-xmin)/n。linespec 用于指定曲线的线型、颜色和标记符号等。

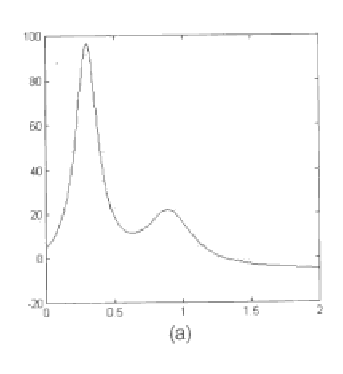
下面一段程序用来调用函数 fplot 绘制曲线：

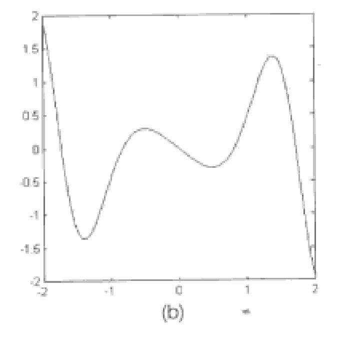
```
f = @(x,n)abs(exp(-1j*x*(0:n-1))*ones(n,1));      % 定义含参数的函数
s(1)=subplot(131);fplot('humps',[0,2]);           % 根据自定义函数绘图
xlabel('(a)','Fontsize',14,'Fontname','Times New Roman');     % X 轴标注
s(2)=subplot(132);fplot(@(x)x.*sin(x.^2-cos(x)),[-2,2],1e-4);   % 根据直接输入的函数绘图
xlabel('(b)','Fontsize',14,'Fontname','Times New Roman');     % X 轴标注
s(3)=subplot(133);fplot(@(x)f(x,8),[0,pi*2],'r:x');           % 绘制含参数的曲线
xlabel('(c)','Fontsize',14,'Fontname','Times New Roman');     % X 轴标注
axis(s,'square');                                             % 设置所有坐标轴为方形
```

这里给出 3 种不同方式的函数定义形式，即调用 M 文件定义的函数（humps.m 函数）、直接输入函数（如图 16.20(b)的绘制）以及预先定义的函数（如图 16.20(c)的绘制）。上述程序执行后得到如图 16.20 所示的图形。



humps 是 MATLAB 自带的一个用函数文件定义的函数。输入函数 fun 可以使用用户自己定义的函数，如用 M 文件定义。对于比较简单的函数表达式，用户还可以利用类似于“f = @(x,n)abs(exp(-1j*x*(0:n-1))*ones(n,1));”的语句来定义。





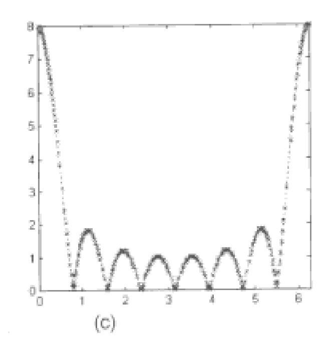


图 16.20　函数 fplot 绘制的曲线

16.1.3.2 输入函数 ezplot 绘图

函数 ezplot 可以根据输入函数进行绘图操作，其调用格式为：

```
ezplot(fun);
ezplot(fun,[a,b]);
```

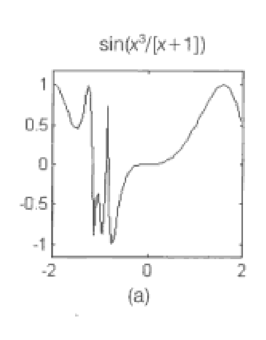
```
ezplot(fun,[xmin,xmax,ymin,ymax])
ezplot(funx,funy,[tmin,tmax])
```

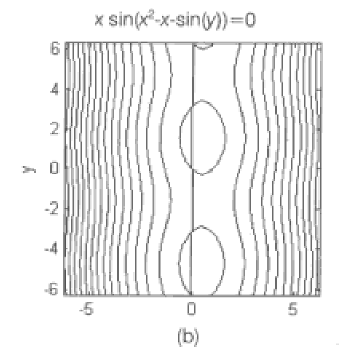
参数说明：fun 是输入函数的名或者表达式，其可以是形如 $f(x)$ 和 $g(x,y)$ 的函数，对于 $g(x,y)$，MATLAB 将计算 $g(x,y)=0$ 对应的曲线。a 和 b 用于指定自变量的取值范围。xmin，xmax，ymin 和 ymax 用于指定横纵坐标的范围。funx 和 funy 是一元函数，tmin 和 tmax 用于设定 funx 和 funy 的自变量范围。

例 16–14：调用函数 ezplot 绘图。

```
g = @(x,n)sin(x.^2+x*n);                          % 定义含参数的函数
s(1)=subplot(141);ezplot('sin(x.^3/[x+1])',[-2,2]);    % 根据自定义函数绘图
xlabel('(a)','Fontsize',14,'Fontname','Times New Roman');  % X 轴标注
s(2)=subplot(132);ezplot(@(x,y)x.*sin(x.^2-x-sin(y)));      % 根据直接输入的函数绘
图
xlabel('(b)','Fontsize',14,'Fontname','Times New Roman');  % X 轴标注
s(3)=subplot(133);ezplot(@(x)g(x,2),@(y)sin(sin(y)*4.5),[0,pi*2]);   % 绘制含参数
的曲线
xlabel('(c)','Fontsize',14,'Fontname','Times New Roman');          % X 轴标注
axis(s,'square');   % 设置所有坐标轴为方形
```

这里给出了 3 种不同输入函数形式的函数绘图示例，同时给出坐标轴范围限制的操作演示。与 plot 函数相比，ezplot 的输入参数要少一些。输出图形如图 16.21 所示，图(a)是一个单值函数曲线，而后面两幅图的函数曲线是多值的，可见利用函数 ezplot 绘制一些多值函数表达式的曲线是很方便的。





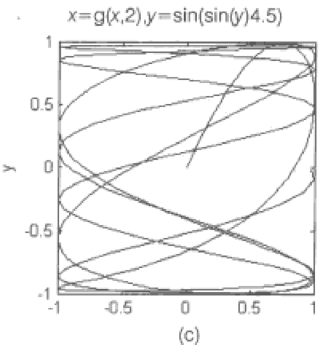


图 16.21 函数 ezplot 绘制的曲线



函数的表达式被显示在坐标轴的图题位置处。

16.1.3.3 用函数表达式绘制等高线图

函数 ezcontour 可以根据函数表达式来绘制等高线图，其调用格式为：

```
ezcontour(fun);
ezcontour(fun,domain);
ezcontour(...,n);
```

参数说明：fun 表示函数的表达式。domain 是一个长度为 2 或者 4 的向量，用于限制横纵坐

标的范围，其默认值为[-pi*2,pi*2]。n 用于指定绘制网格时采用的离散取样点数，其默认值为 60。

例 16-15：调用函数 ezcontour 绘制等高线。

```
g1 = @(x,y)x.^2+y.^3;                          % 定义第一个二元函数
g2 = @(x,y)sin(x.^2)+cos(y.^3);                % 定义第二个二元函数
g3 = @(x,y,c)cos(x.^2+c*y.^2);                 % 定义第三个二元函数
s(1)=subplot(141);ezcontour(@(x,y)g1(x,y),[-2,2]);          % 绘制等高线图
xlabel('(a)','Fontsize',14,'Fontname','Times New Roman');   % X 轴标注
s(2)=subplot(132);ezcontour(@(x,y)g2(x,y),[-2,2],[-3,3]);   % 绘制等高线图
xlabel('(b)','Fontsize',14,'Fontname','Times New Roman');   % X 轴标注
s(3)=subplot(133);ezcontour(@(x,y)g3(x,y,0.8),40);          % 绘制等高线图
xlabel('(c)','Fontsize',14,'Fontname','Times New Roman');   % X 轴标注
axis(s,'square');                                           % 设置所有坐标轴为方形
```

在例子中，使用 3 个不同的二元函数作为输入，并使用了 3 种不同输入方式利用 ezcontour 来绘制等高线图。输出的图形如图 16.22 所示，可见这 3 幅等高线图的等高线疏密程度逐渐变得密集，同时它们还存在着一定的对称性（这与输入函数有关）。

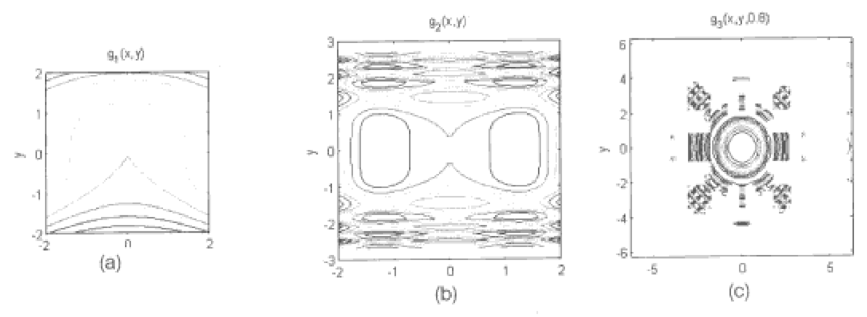


图 16.22　函数 ezcontour 绘制的等高线图



调用函数 ezcontour 时，如果用户使用语句"ezcontour('g1',[-2,2]);"绘制等高线图，MATLAB 将把"g_1"作为自变量来绘制等高线图，这时得到的等高线是一些竖线，而不是用户期望的结果。

16.1.3.4　彩色等高线图

函数 ezcontourf 绘制的等高线在不同区域填充不同的颜色，其调用格式和函数 ezcontour 相同。把上面例子中的"ezcontour"换为"ezcontourf"即可得到如图 16.23 所示的图形。

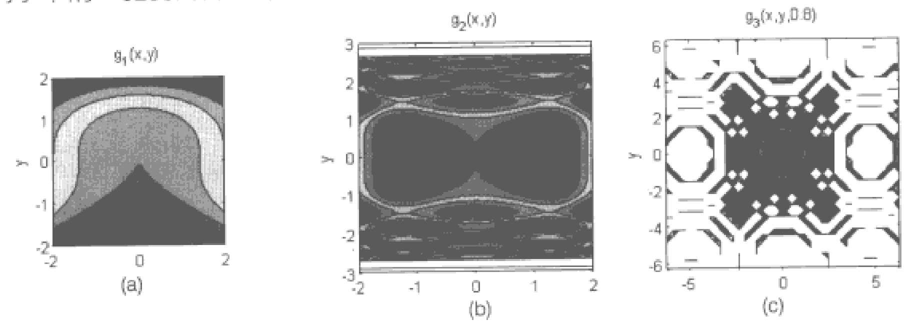


图 16.23　函数 ezcontourf 绘制的彩色等高线图

16.1.3.5 ezpola 极坐标图

函数 ezpolar 可以绘制极坐标图，其调用格式为：

```
ezpolar(fun);
ezpolar(fun,[a,b]);
```

参数说明：fun 是极坐标函数，即 $\rho = \mathrm{fun}(\theta)$。[a,b]用于指定 θ 的范围，其默认值为[0,pi*2]。

例 16-16：调用函数 ezpolar 绘制极坐标曲线。

```
fun=@(t,c)sin(c*t).*sin(3*c*t);
subplot(121);ezpolar(@(t)t.*sin(t/2));                          % 绘制极坐标曲线
xlabel('(a)','Fontsize',14,'Fontname','Times New Roman');          % X轴标注
subplot(122);ezpolar(@(x)fun(x,1/sqrt(3)),[0,pi*4]);               % 绘制极坐标曲线
xlabel('(b)','Fontsize',14,'Fontname','Times New Roman');          % X轴标注
```

输出图形如图 16.24 所示，其中图(a)类似一个心形，而图(b)的形状很像一只蝴蝶。

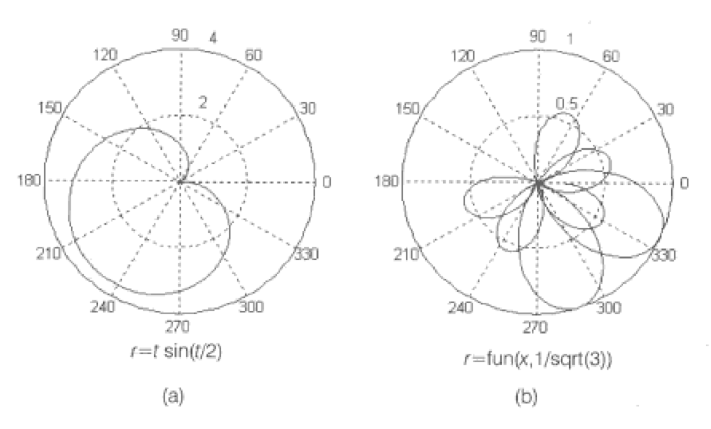


图 16.24 函数 ezpolar 绘制的极坐标曲线

16.2 图形编辑

图形编辑的使用对于修正图形的外观具有重要作用。有时 MATLAB 默认情况下得到的图形可能不能满足用户的要求，此时利用图形编辑功能可以改变现有图形的样式而满足用户的需要。MATLAB 的图形编辑功能可以通过交互式操作，也可以通过程序来实现，二者各自具有自己的优点。本节详细介绍 MATLAB 的图形编辑功能。

16.2.1 应用句柄

句柄（handle）犹如图形对象的标签，通过句柄用户可以改变图形对象的属性值进而改变图形，句柄是一系列 double 型数据。不同对象的句柄不能重复和混淆使用。在图形对象的结构中，各个图形对象之间是分等级的，它们之间的关系如图 16.25 所示。

在图 16.25 中，根（Root）对象是与计算机屏幕有关的一个图形对象。MATLAB 系统只有一个

根对象，它是最上级的对象（上面没有父对象），它的子（Children）对象是 Figure。在 MATLAB 启动的时候，根对象已经被创建了，用户无法重新创建或者删除根对象。根对象的句柄值为 0，可以通过 get(0)来读取该句柄的内容，通过 set(0,…)设定该句柄的属性。该句柄的常用属性如表 16.5 所示。

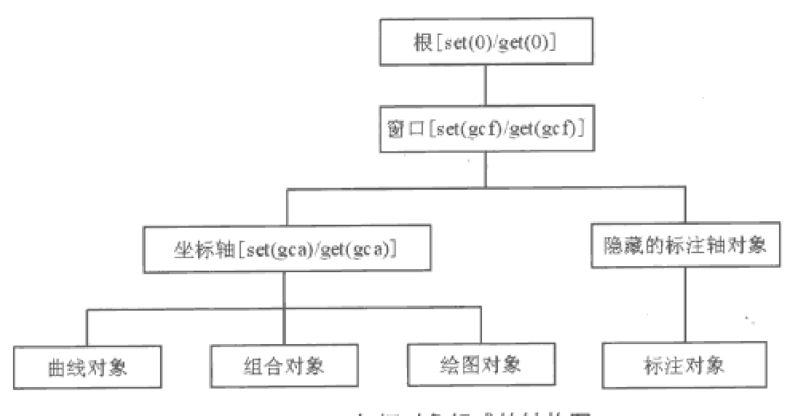


图 16.25　句柄对象组成的结构图

表 16.5　根对象句柄的常用属性

属性名	说明	属性名	说明
CurrentFigure	标识当前图形窗口数目	RecursionLimit	限定递归的最大次数,默认值为 500
Language	系统语言环境，与操作系统有关	PointerLocation	指针的当前位置
ScreenSize	屏幕大小，由[L, B, W, H]定义	Children	子句柄对象,即图形窗口的句柄

Figure 对象（图形窗口）是 MATLAB 中包含 GUI 设计编辑窗在内的一个显示窗口。在系统极限条件限制下，用户可以创建任意个 Figure。如果当前没有 Figure 对象，用户在调用绘图函数（如 plot）时，MATLAB 会自动创建一个 Figure。如果当前存在 Figure，绘图函数将把图形绘制在当前 Figure 上。当前 Figure 的默认句柄是 gcf，用户可以通过函数 get 和 set 读取和设置当前 Figure 的属性。用户还可以通过语句"h=figure"来记录 Figure 的句柄。用户在命令窗中输入 set(gcf)可以得到全部句柄属性及可能取值（其中花括号内的是默认值）。表 16.6 给出了常用图形窗口的句柄属性说明。

表 16.6　常用图形窗口的句柄属性

属性名	说明	属性名	说明
Color	背景颜色	Position	指定图形窗口的位置和大小
DoubleBuffer	控制动画渲染时快速缓冲	Renderer	设置屏幕和图片的渲染方式
MenuBar	控制菜单是否显示	Units	设置显示度量的单位

坐标轴是位于 Figure 上的子对象，一个 Figure 可以有多个坐标轴。坐标轴的创建可以用函数 axes 来创建，其调用格式为：

```
axes('position', rect)
```

参数说明：position 是坐标轴的位置属性。参数 rect 是对应的位置信息，一个由 4 个元素组成

的向量，即 rect = [left, bottom, width, height]，left 和 bottom 是坐标轴左下角点的位置坐标（相对于整个 Figure 空白区域的比例），width 和 height 是坐标轴的宽和高。rect 的默认值为[0.1300, 0.1100, 0.7750, 0.8150]。

当前坐标轴的句柄可以用 gca 表示，用户还可以通过下面两种方式记录句柄：

```
Ax1=axes(...);
Ax2=subplot(...);
```

与 Figure 相似，如果不存在 Figure 或者当前 Figure 上没有坐标轴时，调用绘图函数时，MATLAB 会自动在 Figure 上生成一个坐标轴。如果绘图时存在坐标轴，MATLAB 将把图形绘制在当前坐标轴上。用户在命令窗中输入 Set(gca)可以得到全部句柄属性及可能取值（其中花括号内的是默认值）。表 16.7 给出了坐标轴常用句柄属性说明。

表 16.7 坐标轴常用句柄属性

属性名	说明	属性名	说明
Box	控制上面和左侧边框是否显示	XDir/YDir/ZDir	坐标轴刻度变化方向
FontAngle	控制文本显示为斜体	XGrid/YGird/ZGrid	对应轴网格线是否开启
FontName	设置字体	XTick/ YTick/ZTick	轴上刻度位置
FontSize	设置字号	XTickLabel/YTickLabel/ZTickLabel	设置坐标轴上刻度的文本内容
FontUnits	设置字体大小的度量		
LineWidth	轴线的宽度	XColor/YColor/ZColor	设置轴的颜色（包括网格颜色）
Position	设置坐标轴位置和大小	XLim/YLim/ZLim	设置坐标轴显示范围
TickLength	刻度线的长度	XScale/YScale/ZScale	数据变化的比例：linear，log
TickDir	选择刻度线向内还是向外	XAxisLocation/YAxisLocation	设置标记刻度的轴的位置

曲线对象的句柄是通过一些绘图函数（如 plot，line，ezplot 等）来记录的，即：

```
h1=plot(...);
h2=line(...);
h3=ezplot(...);
```

很多绘图函数都支持类似上面的方式来标记句柄。用户可以通过句柄来改变曲线的数据、线宽、颜色以及标记符号等属性取值。以 plot 函数的输出句柄为例，表 16.8 给出了常用属性说明。

表 16.8 曲线句柄的常用属性

属性名	说明	属性名	说明
Color	设定曲线颜色	MarkerSize	标记符号的大小
EraseMode	绘制或者擦除曲线的模式	MarkerEdgeColor	标记符号边缘的颜色
LineStyle	线型	MarkerFaceColor	标记符号内部区域的颜色
LineWidth	线宽	XData/ YData/ ZData	曲线对应的坐标轴数据
Marker	标记符号		

组合对象是一些由曲线组成的对象，如函数 patch，rectangle 和 area 等绘制的图形，它们也支持输出句柄的形式，即 h=functionname(...)的形式。对于一些绘制图像的函数(如 image，imshow，imagesc 等)，所得的对象（绘图对象）也支持句柄输出。标注对象指由函数 text，xlabel/ylabel/zlabel，

title 以及 legend 等绘制的标注内容，它们也支持句柄输出。通过句柄用户可以改变其中的字体名称、字号、斜体以及颜色等属性。

利用句柄编程修改图形具有可移植性、重复性好、准确性高等优点，其缺点就是确定一些属性值时可能需要花费很多时间，比如对象的位置。

16.2.2 鼠标控制

在图形修改时利用鼠标进行人机交互操作具有快速、方便的特点。本小节介绍这方面的相关知识。

16.2.2.1 确定标注字符串的位置

函数 gtext 可以实现用鼠标确定标注字符串的位置，其调用格式为：

```
gtext('string');
gtext(...,'propertyname',propertyvalue,...);
```

参数说明： string 是待标注的字符串。PropertyName 是属性名。PropertyValue 是属性取值，其可以多组成对出现，用于修饰标注内容的字体和字号等信息。

执行下面语句，可以利用鼠标指定标注内容的位置：

```
gtext('The quick brown fox jumps over the lazy dog.');
gtext('\it\bfWelcome to Beijing','Fontsize',22,'Fontname','Times New
Roman','Rotation',20)
```

执行上面的语句后，会在坐标轴窗口出现一个“十字叉丝”，鼠标可以移动叉丝的交点位置，单击左键后标注的字符就从鼠标单击位置开始输入到坐标轴上。结果如图 16.26 所示。



这里对“Welcome to Beijing”字符串设置斜体、粗体、字号、字体和旋转角度。

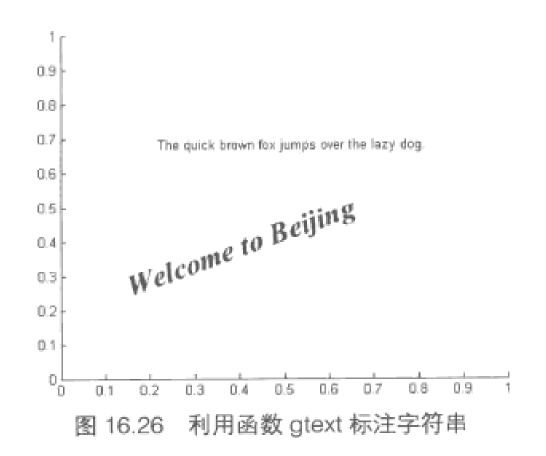


图 16.26 利用函数 gtext 标注字符串

16.2.2.2 从坐标轴上取点的坐标

函数 ginput 可以利用鼠标从坐标轴上取点的坐标，其调用格式为：

```
[x,y] = ginput(n);
```

参数说明：x 和 y 分别是用鼠标取得的横、纵坐标值。n 用于指定取点的个数，当其缺省时，取点数目不受限制，直到用户按下回车键为止。

执行下面的程序即可进行鼠标取点：

```
stem(1:5)          % 绘制针状图
[x,y]=ginput(5);    % 设定鼠标取出 5 个点的坐标
[x';y']             % 显示坐标值
```

执行上述程序可以得到的结果如图 16.27 所示，依次单击针状图的圆圈可以得到相应位置的坐标。输出坐标的数据为：

```
ans =
    1.0046    2.0000    3.0046    4.0092    5.0046
    1.0015    1.9810    2.9751    3.9985    4.9781
```

可见这些坐标的数据是圆圈附近的点。

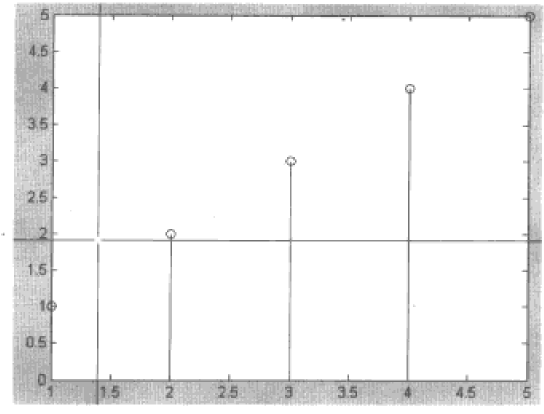


图 16.27　利用函数 ginput 鼠标取点



执行过程中出现的“十字叉丝”如图 16.27 所示，叉丝长度覆盖整个 Figure 的空白区域。

下面介绍利用鼠标改变坐标轴大小的方法。首先单击工具条中的可编辑按钮（Edit plot，如图 16.28(a)所示）。再激活当前的坐标轴，坐标轴边框上会出现 8 个黑色的实心方块，如图 16.28(b)所示。用左键按住坐标轴角点的黑色方块可以按比例缩放坐标轴大小（如图 16.28(c)所示）。如果按住坐标轴中各边中点的黑色方块可以沿相应方向缩放坐标轴（如图 16.28(d)所示）。利用鼠标控制坐标轴大小的优点是比较方便，但是精度控制比较难，批量生成图形时不方便。如果利用语句实现，可以用函数 axes 设定坐标轴的位置。

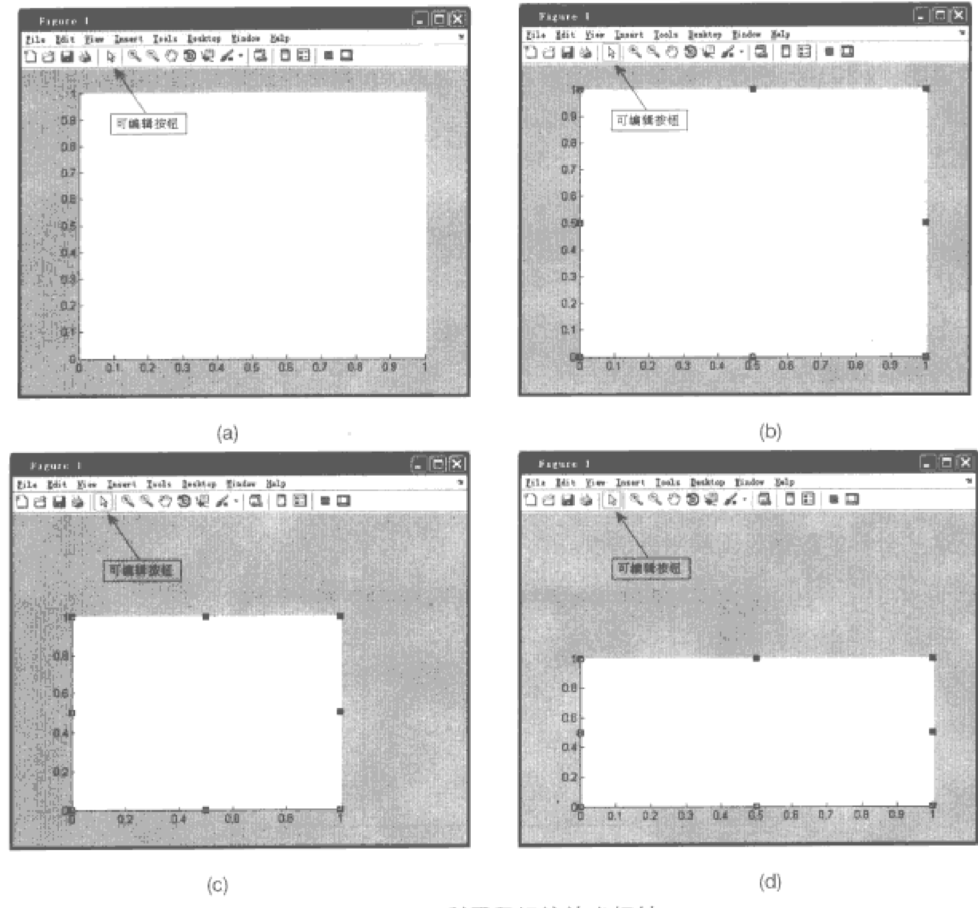


图 16.28　利用鼠标缩放坐标轴

16.2.3　图形注释

本小节来介绍图形注释的一些方法，MATLAB 提供了函数 text、title、xlabel (ylabel 和 zlabel)、legend 等来完成注释的功能。下面依次介绍这些函数的使用。

16.2.3.1　在坐标轴窗口标注

函数 text 可以用来在坐标轴窗口标注内容，其调用格式为：

```
text(x,y,'string');
text(x,y,z,'string');
text(...,'propertyname',propertyvalue,...);
```

参数说明： x，y 和 z 分别为标注内容的起始位置在 X 轴、Y 轴和 Z 轴的刻度。string 代表标注的文本内容。PropertyName 和 PropertyValue 是标注内容属性名称和取值。

16.2.3.2　在图题位置标注

函数 title 用来在图题位置处标注，用于指定图形中曲线或者图形的内容，其调用格式为：

```
title('text');
title('text','property1',propertyvalue1,'property2',propertyvalue2,...)
```

参数说明: text 代表标注的文本内容。property1(property2)和 propertyvalue1(propertyvalue2)等分别为标注内容属性名称和取值。

16.2.3.3 坐标轴的标注

函数 xlabel，ylabel 和 zlabel 可分别用于在坐标轴的 X 轴、Y 轴和 Z 轴标注，这 3 个函数用法相同。这里以函数 xlabel 为例介绍其使用方法，调用格式为：

```
xlabel('text')
xlabel('text','property1',propertyvalue1,'property2',propertyvalue2,...)
```

参数说明: text 代表标注的文本内容。property1(property2)和 propertyvalue1(propertyvalue2)等分别为标注内容属性名称和取值。

16.2.3.4 标注图例

函数 legend 可用于指定每条曲线对应的内容（即图例），其调用格式如下：

```
legend(string1,string2,string3, ...); % 格式 1
legend(h,string1,string2,string3, ...); % 格式 2
legend off 或 legend(ax,'off');
legend hide 或 legend(ax,'hide');
legend show 或 legend(ax,'show');
legend boxoff 或 legend(ax,'boxoff');
legend boxon 或 legend(ax,'boxon');
legend(...,pos);
```

参数说明: string1，string2，string3 等表示标注中的字符串。在格式 1 中，MATLAB 自动找到坐标轴中的曲线并对曲线线型、线宽、颜色、标记符号等加以标注，当标注内容（string1，string2，string3…）少于曲线的条数时，MATLAB 根据标注字符串数目选取曲线；如果标注内容（string1，string2，string3…）超过曲线的条数，多余的标注字符串中对应的曲线样段部分将是空白。在格式 2 中，可以对指定句柄 h 中的曲线进行注释，而该句柄外曲线将不予标注。对于格式 2，可以选用格式 1 等价替换，即先画出需要标记的曲线并用函数 legend 标注，不想标注的曲线在之后用函数 legend 绘制。参数 off 用于关闭标注内容。hide/show 用于隐藏/显示标注内容。boxoff/boxon 用于关闭/打开标注图例部分之外的边框。pos 用于指定图例相对于坐标轴的位置，其可选值为-1，0，1，2，3，4。各数值的含义如下：-1 表示坐标轴外面的右上角，0 表示自动选择一个最好位置（即图例覆盖曲线最少），1 表示右上角（为默认值），2 表示左上角，3 表示左下角，4 表示右下角。

16.2.3.5 示例说明

下面结合本小节介绍的各个函数进行图形注释给出相应的例子，程序内容如下：

```
figure('Position',[57 99 891 538]);        % 生成图形窗口并指定位置
t=linspace(0,pi,101);                      % 生成采样数据
s1=subplot(121);
hp=plot(t,sin(2*t),'r:',t,cos(t*2),'k');    % 绘制曲线
```

```
hold on;
plot(t(1:5:end),sinc(t(1:5:end)),'k-o','markersize',4);    % 绘制 sinc 函数曲线
set(s1,'Position',[0.13 0.31 0.327023 0.415],'xlim',[min(t),max(t)]);
text(0.1,-0.4,'This is the sine and cosine functions.','Rotation',20);    % 标注文本字符
L1=legend(hp,'sin(2{\itt})','cos(2{\itt})',3);                        % 曲线的标签
xlabel({'{\itt}','(a)'},'Fontsize',14,'Fontname','Times new roman');    % 标注 X 轴注释内容
ylabel('AF','Fontsize',14,'Fontname','Euclid Math one');             % 标注 Y 轴注释内容
title('Function curves','Fontname','Monotype Corsiva','Fontsize',16);  % 标注图题
s2=subplot(122);
plot3(cos(t*8),sin(t*8),t,'k-o');                                 % 绘制螺旋曲线 1
hold on;
plot3(cos(t*8).*t/max(t),sin(t*8).*t/max(t),t,'r-x');                 % 绘制螺旋曲线 2
xlabel('{\itx}({\itt})','Fontsize',14,'Fontname','Times new roman');    % 标注 X 轴注释内容
ylabel('{\ity}({\itt})','Fontsize',14,'Fontname','Times new roman');    % 标注 Y 轴注释内容
zlabel('{\itz}({\itt})','Fontsize',14,'Fontname','Times new roman');    % 标注 Z 轴注释内容
text(-0.5,0,5,'helix','Fontsize',16,'Fontname','Times new roman');     % 用函数 text 标注图题
text(-0.5,0,-2.65,'(b)','Fontsize',14,'Fontname','Times new roman');    % 用函数 text 标注图题
set(s2,'Position',[0.577977 0.31 0.327023 0.515]);                   % 设定坐标轴位置
L2=legend('Helix1','Helix2',2);                                   % 曲线的标签
set([L1,L2],'Fontname','Times new roman');                          % 设置两个图例标注的字体
```

这里给出 sin，cos，sinc 函数的绘制（如图 16.29(a)所示），同时给出等距螺旋线的绘制（如图 16.29(b)所示），其中使用了不同的线型和标记符号。需要注意的是标注性文字应该离开曲线一段距离，尽量不要覆盖曲线。

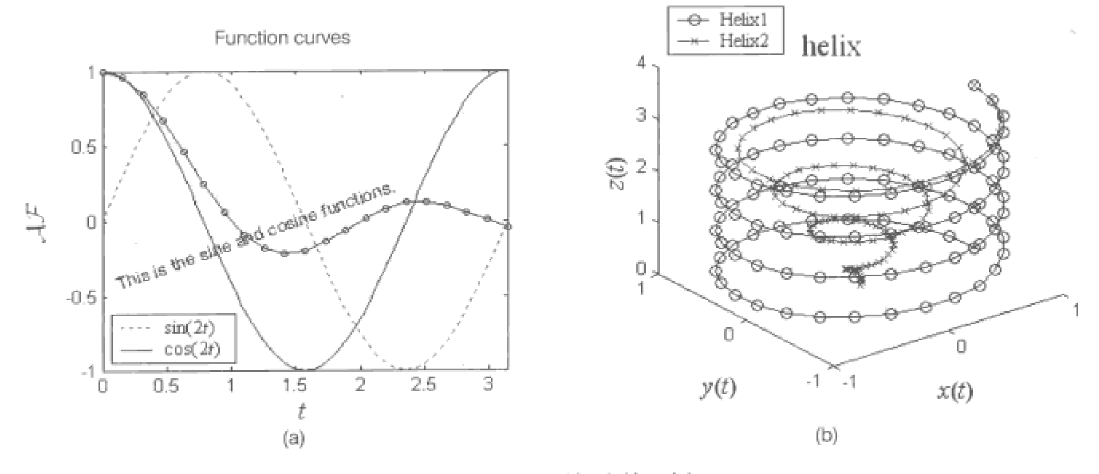


图 16.29　图形标注的示例

16.2.4 字体设定

选择精美的字体对于表现图形具有重要的辅助作用，用户可以通过 Office 软件来查阅字体样式。在标注函数的 FontName 项中设定选择的字体即可。在一些公式中经常用到希腊字母和特殊符号，表 16.9 列出希腊字母和一些数学符号的输入方法。该表是利用 MATLAB 绘制的，相应程序保存在光盘中的\Ch16 文件夹中，文件名是 Greek_Letters.m。用户把控制符写入到注释函数的 string 部分就可以显示相应的“输出字符”了。

表 16.9 希腊字母和一些数学符号

控制符	输出字符	控制符	输出字符	控制符	输出字符	控制符	输出字符
\alpha	α	\beta	β	\gamma	γ	\delta	δ
\epsilon	ε	\zeta	ζ	\eta	η	\theta	θ
\vartheta	ϑ	\iota	ι	\kappa	κ	\lambda	λ
\mu	μ	\nu	ν	\xi	ξ	\pi	π
\rho	ρ	\sigma	σ	\varsigma	ς	\tau	τ
\equiv	≡	\Im	ℑ	\otimes	⊗	\cap	∩
\supset	⊃	\int	∫	\rfloor	⌋	\lfloor	⌊
\perp	⊥	\wedge	∧	\rceil	⌉	\vee	∨
\langle	〈	\upsilon	υ	\phi	ϕ	\chi	χ
\psi	ψ	\omega	ω	\Gamma	Γ	\Delta	Δ
\Theta	Θ	\Lambda	Λ	\Xi	Ξ	\Pi	Π
\Sigma	Σ	\Upsilon	ϒ	\Phi	Φ	\Psi	Ψ
\Omega	Ω	\forall	∀	\exists	∃	\ni	∋
\cong	≅	\approx	≈	\Re	ℜ	\oplus	⊕
\cup	∪	\subseteq	⊆	\in	∈	\lceil	⌈
\cdot	·	\neg	¬	\times	×	\surd	√
\varpi	ϖ	\rangle	〉	\sim	∼	\leq	≤
\infty	∞	\clubsuit	♣	\diamondsuit	♦	\heartsuit	♥
\spadesuit	♠	\leftrightarrow	↔	\leftarrow	←	\uparrow	↑
\rightarrow	→	\downarrow	↓	\circ	°	\pm	±
\geq	≥	\propto	∝	\partial	∂	\bullet	•
\div	÷	\neq	≠	\aleph	ℵ	\wp	℘
\oslash	∅	\supseteq	⊇	\subset	⊂	\o	ο
\nabla	∇	\ldots	…	\prime	′	\O	Ø
\mid	|	\copyright	©				

16.3 自定义特殊图形样式

本节进一步介绍一些特殊图形样式的绘制，通过这些修饰可以美化图形。下面来具体介绍一些方法。

16.3.1 用特殊字符标注刻度

一般情况下，坐标轴的刻度是数字形式的。可以通过程序把刻度变为一些字符形式。下面举例说明实现方式。实现程序如下：

```
figure;                      % 生成一个新的图形窗口
axes;                        % 生成一个空的坐标轴
set(gca,'xtick',0:.2:1);     % 设置刻度位置
```

```
xL={'0','\pi','2\pi','3\pi','4\pi','5\pi'};      % 设置标注内容的字符
set(gca,'XTickLabel',[],'Fontname','Times New Roman','Fontsize',12,'');  % 删去刻度值
box on;                       % 保留上侧和右侧的边框
for k=1:6;

text([k-1]/5,min(ylim)-abs(diff(ylim))/30,xL(k),'HorizontalAlignment','center',...
      'Fontname','Times New Roman','Fontsize',12);   % 利用函数 text 标注新刻度值
End
```

这段程序中给出了坐标轴刻度修改的示例，其中删去了原来的刻度值而填上了新的标注刻度，输出图形如图 16.30 所示。其实现思想是利用函数 text 在横轴相应位置标注上需要的字符。

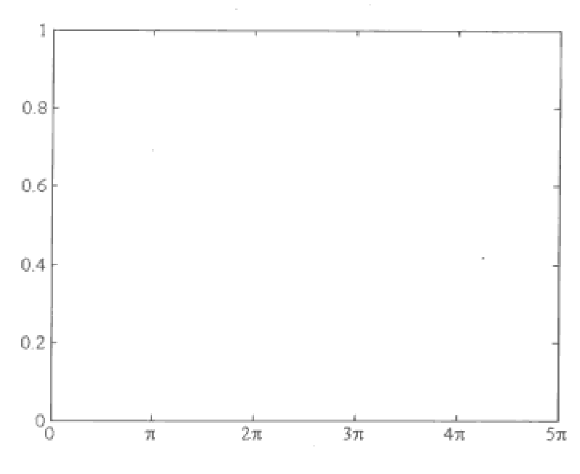


图 16.30 修改坐标轴的刻度值

16.3.2 用特殊图案填充条状图

对于函数 bar 绘制的条状图，利用颜色可以很好地区分不同竖条。如果用户受限制不能使用彩色来区分，可以考虑使用不同的填充图案来完成这个任务。这里考虑新建函数文件来实现这样的功能，条状图可以认为是由多个形如"Π"的图形构成的，填充内容可以通过不同符号来完成。根据上面分析编写函数文件 barfill.m 实现上述功能（该文件保存在光盘的\Ch16 文件夹下）。该函数调用格式为：

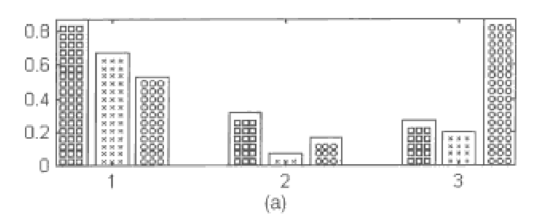
```
barfill(y,mark,bw);
```

参数说明： y 是待分析的数据。mark 是一个用于指定标记符号的字符串，如表 16.2 中第 3 列所示的符号，mark 的长度应该不小于 y 的列数。bw 用于指定条状图的宽度。

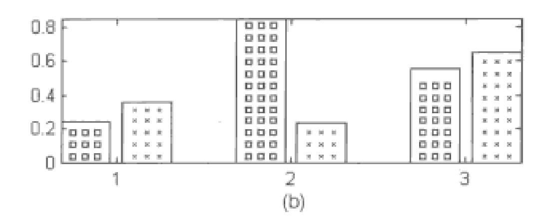
调用函数 barfill 绘制字符串，相应程序如下：

```
subplot(121);
barfill(rand(3,3),'sxo',0.7);  % 用符号填充的图形
axis image;xlabel('(a)');     % 设置坐标轴长宽比例，同时标注 x 轴
subplot(122);
barfill(rand(3,2),'sxo',0.7);  % 用符号填充的图形
axis image;xlabel('(b)');     % 设置坐标轴长宽比例，同时标注 x 轴
```

所得图形如图 16.31 所示，这里以服从均匀分布随机数作为输入数据，利用函数 barfill 给出了两种填充示例。



(a)



(b)

图 16.31 用符号填充的条状图

16.3.3 自定义网格

MATLAB 中可以使用语句 grid on 来显示网格，实际上这些网格是两组平行于 X 轴和 Y 轴的虚线，可以利用函数 plot 自行绘制一些这样的线以便按要求添加网格线。这里编写函数文件 plotg.m（该文件保存在光盘的\Ch16 文件夹下）实现网格线的绘制，该函数的调用格式为：

```
plotg(x,y,d);
```

参数说明： x 和 y 是绘图数据，表示平行于 X 轴或者 Y 轴曲线的端点。d 是一个字符串，用于指定绘制 X 轴还是 Y 轴上的网格线，可取值为 x 或者 y。

使用语句 grid on 和函数 plotg 来绘制不同形式的网格线，相应程序如下：

```
t=linspace(0,pi*2,201);   % 生成离散数据
s(1)=subplot(231);
plot(t,sin(t));  % 绘图
grid on;      % 显示网格
xlabel('(a)','Fontsize',12);        % X 轴标注
s(2)=subplot(232);plot(t,sin(t));  % 绘图
set(gca,'XGrid','on');            % 显示 X 轴上的网格
xlabel('(b)','Fontsize',12);        % X 轴标注
s(3)=subplot(233);plot(t,sin(t));  % 绘图
set(gca,'YGrid','on');            % 显示 Y 轴上的网格
xlabel('(c)','Fontsize',12);        % X 轴标注
s(4)=subplot(234);plot(t,sin(t));  % 绘图
plotg([1:.3:2.1,4:.3:5.5],ylim,'x');
xlabel('(d)','Fontsize',12);        % X 轴标注
s(5)=subplot(235);plot(t,sin(t));  % 绘图
plotg(xlim,[-0.3:.15:0.3],'y');
xlabel('(e)','Fontsize',12);        % X 轴标注
s(6)=subplot(236);plot(t,sin(t));  % 绘图
plotg(xlim,[-0.3:.15:0.3],'y');
plotg([1:.3:2.1,4:.3:5.5],ylim,'x');
grid on;                  % 显示网格
xlabel('(f)','Fontsize',12);      % X 轴标注
```

这段程序给出了 MATLAB 中自带的网格线与函数 plotg 绘制的网格线的比较，利用函数 plotg 可以在任意位置添加网格线。输出图形如图 16.32 所示，其中图(a)、图(b)和图(c)给出的网格线是等间隔的，而图(d)、图(e)和图(f)给出的网格线是指定范围的等高线且可以不等间距。

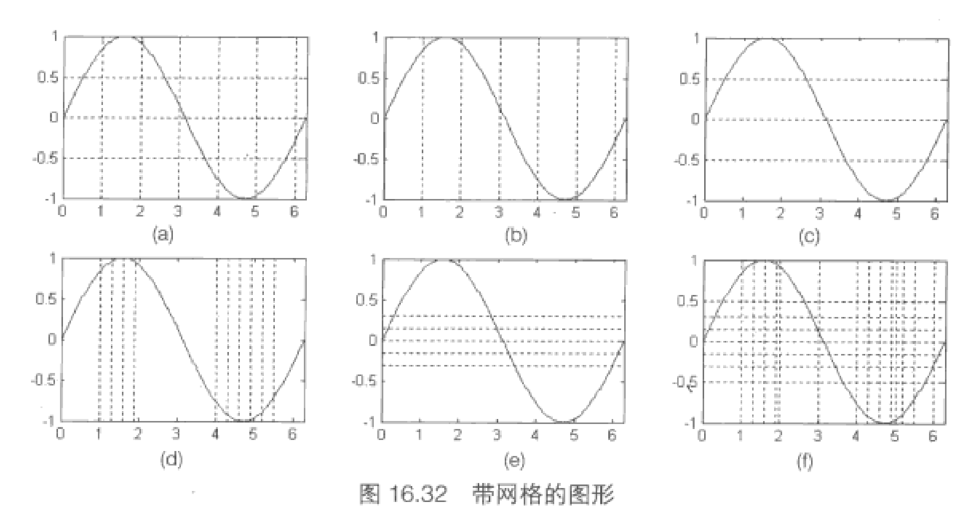


图 16.32　带网格的图形



利用函数 plotg 可以在任意位置绘制刻度线，对于图形中曲线各点的数值请查阅有关帮助。

16.3.4　画箭头

MATLAB 中箭头的绘制可以用下面的形式以字符形式标注：

```
text(x,y,'\leftarrow');
text(x,y,'\rightarrow');
text(x,y,'\uparrow');
text(x,y,'\downarrow']);
```

箭头的大小可以通过 Fontsize 属性来设置。上述语句得到的箭头形式为←，→，↑，↓。上述箭头是文本格式的，用户还可以绘制一般格式的箭头，从 MathWorks 网站上可以下载到相应的程序 arrow.m，下载网址是：http://www.mathworks.com/matlabcentral/ fileexchange/loadFile.do?objectId=278&objectType=File。该函数的调用格式为：

```
arrow(start,stop)
arrow(...,'property1',propval1,'property2',propval2,...)
```

参数说明： start 表示起点的坐标值，二维情况时是1×2 的向量，三维情况时是1×3。stop 是终点的坐标。property1 和 property2 等是相关属性的名称，propval1 和 propval2 等是属性对应的取值。该函数的常用属性有：length 表示箭头的长度，baseangle 表示箭头尾部的角度，tipangle 表示箭头头部半角宽，width 表示箭柄的线宽，如图 16.33 所示。

上面介绍的 arrow 函数绘制的箭头是在坐标轴上的，而且本章所介绍的绘制函数所绘制的图形都限制于坐标轴范围内。在 MATLAB 中还提供了函数 annotation 实现在 Figure 窗口绘制箭头，其调用格式为：

```
annotation('arrow',x,y);
```

参数说明： arrow 表示绘制单头箭头（该函数还可以绘制双头箭头、矩形、椭圆、文本框、线段、文本箭头）。x 和 y 表示箭头的两个端点的坐标值（坐标值是相对 Figure 窗口而言的，其中左

下角点坐标值为(0,0)，右上角点坐标值是(1,1))。

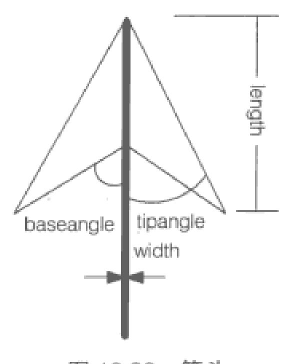


图 16.33 箭头

下面调用上面介绍的函数绘制不同的箭头和一些特殊图形，实现程序如下：

```
T=text(0.5,0.5,['\leftarrow','\rightarrow','\uparrow','\downarrow'],'Fontsize'
,14,'Units','normalized');
axis([0,8,0,8]);                        % 设置坐标轴显示范围
arrow([1,2],[3,4],'Length',20,'BaseAngle',30,'TipAngle',20,'Width',2);  % 绘制坐
标轴内的箭头
annotation('arrow',[0.28,0.21],[0.86,0.95]);  % 在 Figure 窗口绘制单头箭头
annotation('rectangle',[0.8,0.8,0.15,0.1]);    % 在 Figure 窗口绘制矩形
annotation('ellipse',[0.8,0.65,0.15,0.1]);     % 在 Figure 窗口绘制椭圆
annotation('textbox',[0.8,0.5,0.15,0.1],'String',{'ABCDEFGH','0123456789'});  %
在 Figure 窗口绘制文本框
annotation('line',[0.8,0.95],[0.46,0.46],'Linewidth',2);        % 在 Figure 窗口绘制
线段
annotation('doublearrow',[0.8,0.95],[0.4,0.4],'Linewidth',2);  % 在 Figure 窗口绘
制双头箭头
annotation('textarrow',[0.8,0.95],[0.32,0.23],'String','Liu','Linewidth',1);  %
文本箭头
```

这里给出利用函数 arrow 和 annotation 绘制箭头的例子，此外利用函数 annotation 还可以绘制其他的基本图形。输出图形如图 16.34 所示，在不同位置处给出了基本图形绘制，读者可以模仿示例用于自己的图形中。

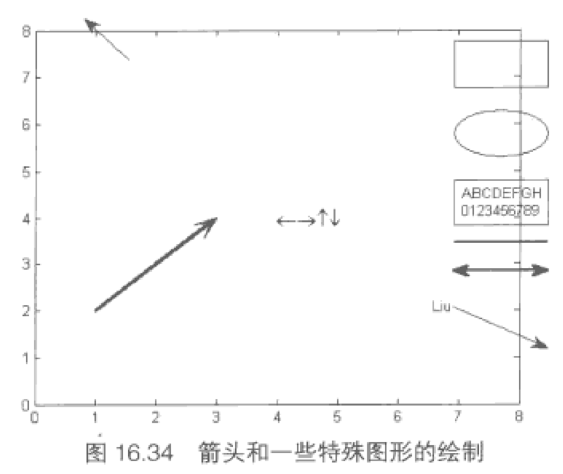


图 16.34 箭头和一些特殊图形的绘制



因为函数 annotation 可以绘制结合文字和箭头的图形，因此它可用来标注一些图形的关键区域。利用函数 text 绘制的箭头并没有因为函数 axis 改变范围而改变在坐标轴内的相对位置。

16.3.5 多值函数的绘制

对于多值函数的绘制可以考虑调用函数 ezplot 进行绘图，下面结合函数 ezplot 绘制下面两个二元函数对应的曲线：

$$x^2+\frac{y^3}{\sqrt[3]{x^2-y^4}}=1\text{，}\quad x^2\operatorname{sinc}(xy)+\frac{y^2}{\sqrt{x^2-y^4}}=2$$

相应实现程序如下：

```
s(1)=subplot(121);
ezplot('x^2+y^3*(x^2-y^4)^[-1/3]-1');                          % 调用函数 ezplot 绘图
s(2)=subplot(122);
ezplot('x^2*sinc(x*y)+y^2*(x^2-y^4)^[-1/2]-2',[-10,10]);  %调用函数 ezplot 绘图，并指明变量取值范围
axis(s,'square');                                              % 设置两个坐标轴为方形
```

输出图形如图 16.35 所示，其中左图相对简单些，右图是一个复杂的多值函数图，同时它们具有一定的对称性。

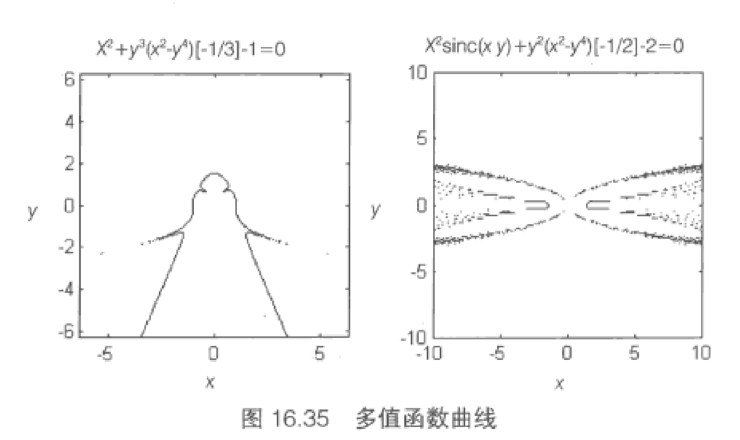


图 16.35　多值函数曲线

16.4 基本图形的绘制

本节来介绍一些基本几何图形的绘制，通过它们的组合，用户可以得到一些复杂的图形。基本的几何图形都可以利用函数 plot 绘制出来，关键问题是确定几何图形的端点坐标值。下面依次介绍基本几何图形的绘制。

16.4.1 线段和弧线

线段只需要两个端点就能确定，而圆弧可以看做是圆的一部分，通过圆心和角度范围就可以确定圆弧。下面结合例子说明它们的绘制。

```
figure;hold on;  % 新建图形窗口
plot([1,3],[1,1],'k');plot([2,2],[3,2],'k');plot([2,3],[4,3],'k'); % 绘制3条线段
plot(3+i+1.2*exp(i*linspace(1,3)),'k');      % 绘制圆弧
plot(3+4i+0.5*exp(i*linspace(0,pi*2)),'k');  % 绘制圆形
axis([0,5,0,5],'equal');  % 设置坐标轴范围和坐标轴长度比例
```

所得图形如图16.36所示，其中给出了线段、圆弧和圆等基本图案的绘制结果。

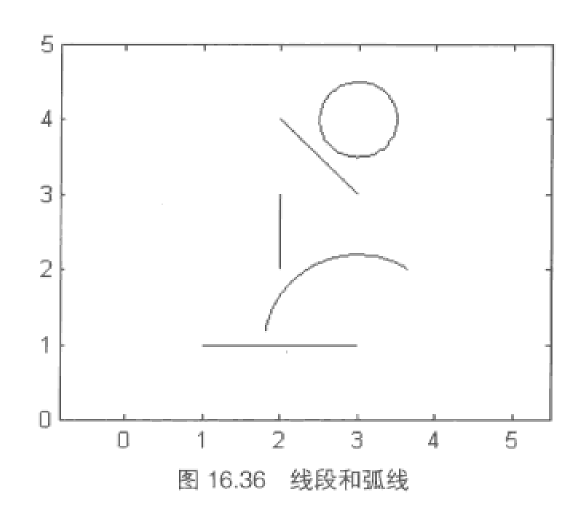


图16.36　线段和弧线

16.4.2 矩形

矩形的绘制可以利用函数plot依次连接4个顶点而成。在MATLAB中提供了函数rectangle专门来绘制矩形，该函数的调用格式为：

```
rectangle('position', [x y w h])
rectangle('curvature', [a b], ...)
```

参数说明：position是一个属性名，用于定义矩形的位置和大小。x和y分别是矩阵左下角点的横、纵坐标值。w和h表示矩形的宽和高。curvature表示矩形4个角点的圆弧大小：当a=b=0（其为默认值）时得到一个矩形；当a=b=1时得到一个椭圆；当a和b介于0~1之间时得到一个圆角矩形。

调用函数rectangle绘制矩形，实现程序如下：

```
axis([0,7,-2,2],'equal');ylim(ylim-0.5) ;hold on;box on;    % 设置坐标轴范围和比例
rectangle('Position',[1,1,2,1]);                          % 绘制矩形
rectangle('Position',[4,1,2,1],'Curvature',[1,1]);          % 绘制椭圆
rectangle('Position',[1,-1,2,1],'Curvature',[0.2,0.5]) ;     % 绘制圆角矩形
plot(5+cos(linspace(0,pi*2)),sin(linspace(0,pi*2))/2-0.5,'k:'); % 绘制椭圆
```

所得图形如图16.37所示。



虚线椭圆是根据椭圆方程 $a^2\cos^2\theta+b^2\sin^2\theta=1$ 绘制的。

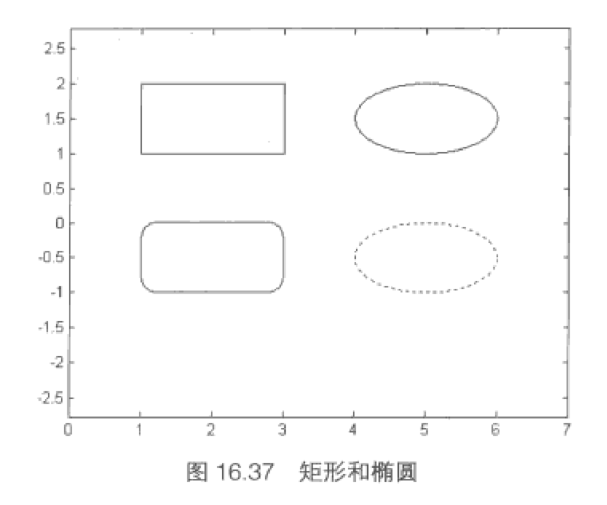


图 16.37　矩形和椭圆

16.4.3　正 N 边形和圆

正 N 边形的绘制可以依次连接所有顶点得到，通用程序可以表示为：

```
plot(exp(i*[linspace(0,pi*2,N+1)+pi/2]),'k');
```

其中 N 表示多边形的边数。利用上面语句绘制正三角形、正六边形、正七边形和正十边形的程序如下：

```
s(1)=subplot(141);N=3;plot(exp(i*[linspace(0,pi*2,N+1)+pi/2]),'k');% 绘制正三角形
s(2)=subplot(142);N=6;plot(exp(i*[linspace(0,pi*2,N+1)+pi/2]),'k');% 绘制正六边形
s(3)=subplot(143);N=7;plot(exp(i*[linspace(0,pi*2,N+1)+pi/2]),'k');% 绘制正七边形
s(4)=subplot(144);N=10;plot(exp(i*[linspace(0,pi*2,N+1)+pi/2]),'k');% 绘制正十边形
axis(s,'square'); % 设置坐标轴为方形
```

图形如图 16.38 所示，图中给出了几种正 N 边形的示例，在图形中央标注了边的数目。

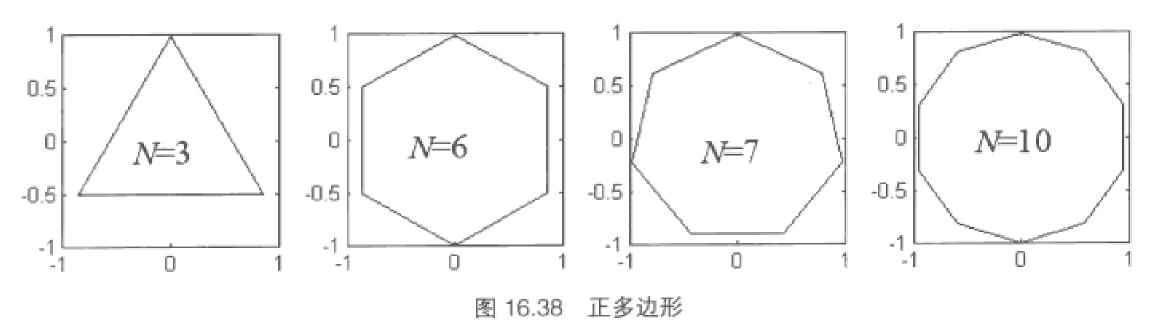


图 16.38　正多边形

在语句“plot(exp(i*[linspace(0,pi*2,N+1)+pi/2]),'k');”中，当 N 取值较大时正多边形就趋近于圆，比如 N=200。用户可以改变 N 的数值尝试。

正 N 角星的绘制思想可以这样设计：当 N 是奇数时，从任意顶点开始按顺时针方向每隔 dn（dn= [N-1]/2）个点连接即可得到 N 角星图案；当 N 是偶数时，任意选择两个相邻的点，分别从这两个点开始按顺时针方向每隔两个点连接起来。根据上面的思想编写 nstar.m 文件（该文件保存在光盘的\Ch16 文件夹下）来绘制 N 角星。该函数调用格式为：

```
nstar(N);
```

参数说明： N 是 N 角星的顶点数。该函数文件的内容不在此显示了，感兴趣的用户可以查看光盘中\Ch16 文件夹下的 nstar.m 文件。

调用函数 nstar 可以得到不同的多角星，下面绘制几个图形作为示例，实现程序如下：

```
subplot(231);nstar(5);axis square;title('{\itN}=5');  % 绘制 5 角星
subplot(232);nstar(9);axis square;title('{\itN}=9');  % 绘制 9 角星
subplot(233);nstar(15);axis square;title('{\itN}=15');  % 绘制 15 角星
subplot(234);nstar(6);axis square;title('{\itN}=6');  % 绘制 6 角星
subplot(235);nstar(8);axis square;title('{\itN}=8');  % 绘制 8 角星
subplot(236);nstar(10);axis square;title('{\itN}=10');  % 绘制 10 角星
```

输出图形如图 16.39 所示，可见其中 N 为奇数时，图形可以一笔画出来，而 N 为偶数时是两个正 N/2 边形的错位叠放。

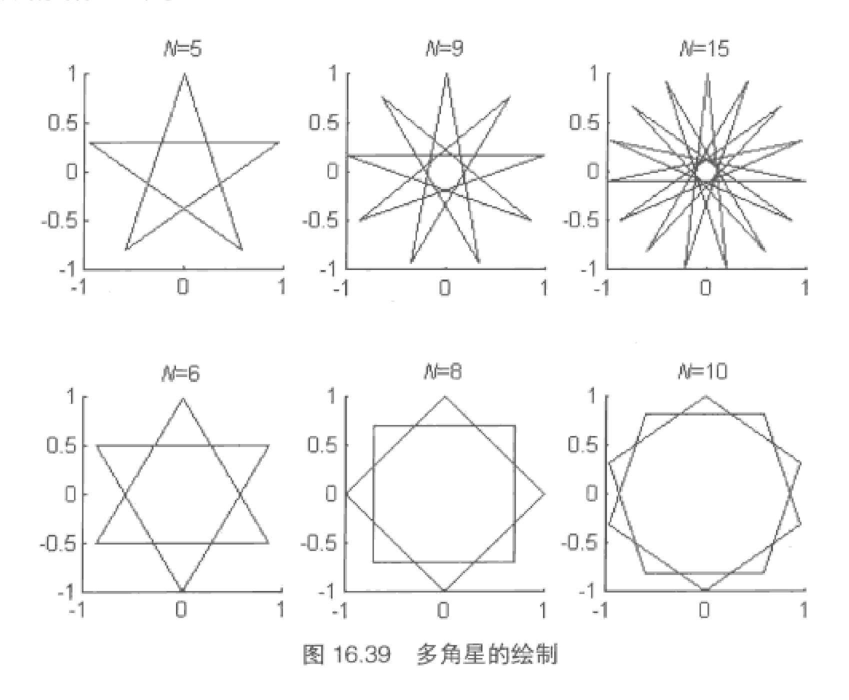


图 16.39 多角星的绘制

16.4.4 弯曲的圆管

把弯曲的圆柱按轴线分若干段，在每个垂直于轴线的断面都是圆，而圆的法线就轴线的切向方向。柱面的绘制可以用若干条平行于轴线的曲线表示。作者根据上述思想编写了相应的程序，将其保存于光盘的\Ch16 文件夹下，文件名为 tube_draw.m。因程序复杂，不在这里显示了。执行该程序后可以得到如图 16.40 所示的图形，其中左图是截面和法线的示意图。

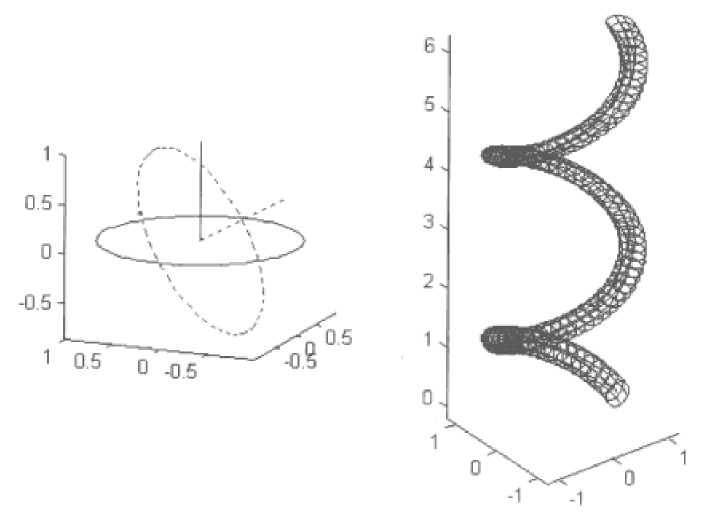


图 16.40　弯曲的圆管

16.4.5　封闭图形的填充

图形填充对部分区域起到突出的作用，可以利用 fill，area，patch 等 3 个函数完成。这 3 个函数的基本调用格式为：

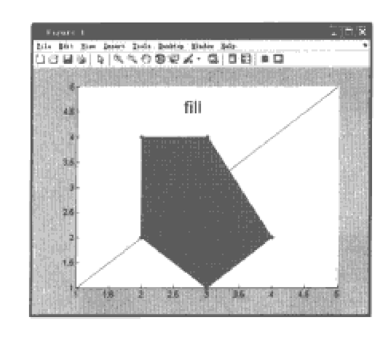
```
fill(x,y,c);
patch(x,y,c);
area(x,y);
```

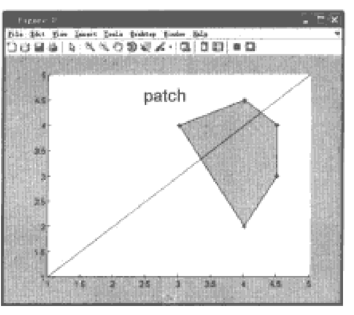
参数说明： x 和 y 是区域顶点的横、纵坐标，要求坐标值按顺序排列。c 用于指定填充的颜色，其颜色可以是定义特殊颜色的字母构成的字符串，也可以是1×3 的向量。

调用这个 3 个填充函数实现填充区域，实现程序如下：

```
figure;hold on;plot([1,5],[1,5],'k');                    % 生成图形窗口并绘制线段
fill([3,2,2,3,4],[1,2,4,4,2],[0.6,0.6,0.6]);             % 填充区域为灰色
plot([3,2,2,3,4],[1,2,4,4,2],'*');                       % 绘制区域顶点
figure;hold on;plot([1,5],[1,5],'k');                    % 生成图形窗口并绘制线段
patch([3,4,4.5,4.5,4],[4,2,3,4,4.5],[0.6,0.6,0.6],'FaceAlpha',0.5); % 填充区域为
灰色，同时指定透明度
plot([3,4,4.5,4.5,4],[4,2,3,4,4.5],'*');                 % 绘制区域顶点
figure;hold on;plot([3,5],[3,5],'k');                    % 生成图形窗口并绘制线段
h=area([3,4,4.5,4.5,4,3],[4,2,3,4,4.5,4]);               % 填充区域
plot([3,4,4.5,4.5,4],[4,2,3,4,4.5],'r*');                % 绘制区域顶点
```

在这个例子中分别利用函数 fill，patch 和 area 来填充指定的区域，而在调用 patch 时进行了透明度设置。上述程序执行后得到如图 16.41 所示的 3 个图形。





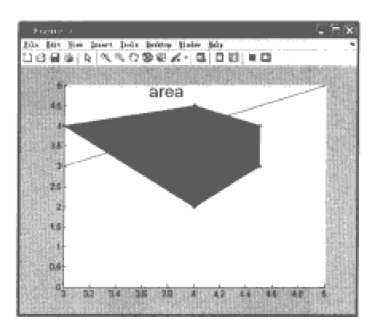


图 16.41 填充区域



利用属性 FaceAlpha 可以改变填充区域的透明度（参见图 16.41 中 patch 函数得到的图形），该属性的默认值为 1（不透明）。函数 area 填充区域的颜色是蓝色，可以通过句柄来改变填充颜色。

16.5 多图布局

有时出于节省空间或者对曲线进行比较等目的，在同一个图形窗口中要设置多个坐标轴。实现这一目的的函数有 subplot 和 axes。下面介绍这两个函数的用法。

16.5.1 subplot 函数

前面程序中已经用到了函数 subplot，这个函数用于生成多个子图，它的调用格式为：

```
subplot(m,n,p)    %格式 1
subplot(mnp)      %格式 2
```

参数说明： m 和 n 分别表示图形窗口中子图的行数和列数。p 表示子图的序号，其值为 1，…，$m\times n$。格式 1 是一种标准写法，而格式 2 是一种省略写法，当 m，n 和 p 都是一位整数时允许这样写，如果它们之中存在一个两位整数，MATLAB 将会出现如下提示：

```
??? Error using ==> subplot at 141
Index must be a 3-digit number of the format mnp.
```

此时换用第一种格式即可。除了按常规调用函数 subplot 绘制 $m\times n$ 的子图阵列外，还可以生成一些特殊排布的子图阵列，比如下面的示例。

```
figure;
subplot(231);subplot(234);  % 生成第一列子图
subplot(233);subplot(236);  % 生成第三列子图
subplot(132);               % 生成另外一种排布方式的第二个子图
figure;
subplot(211);               % 生成第一个子图
subplot(223);subplot(224);  % 生成另外一种排布方式的第二行子图
```

执行上述程序后可以得到如图 16.42 所示的多图布局形式。

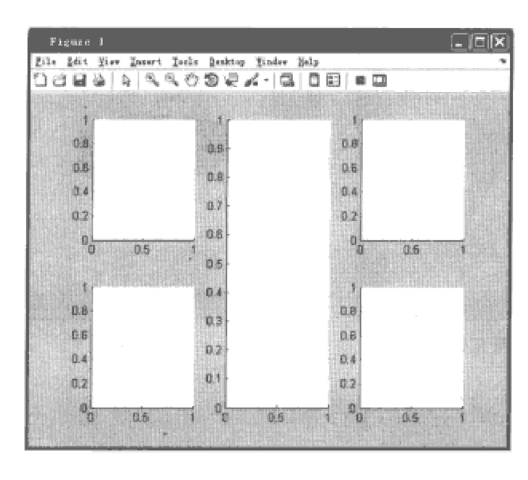
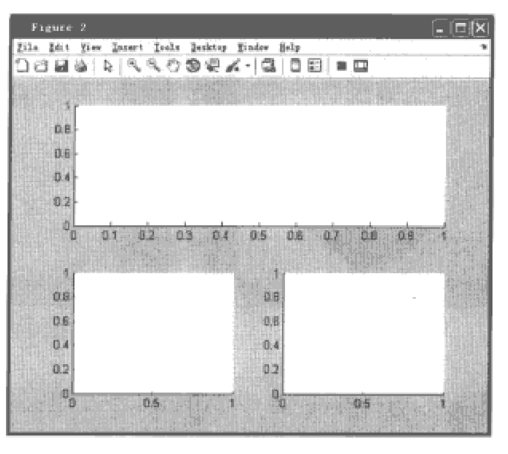



图 16.42　多图布局形式



上面程序设计思路是把两个或者多个子图的位置替换为一个较大的子图，用户在相应子图后面写入绘图语句就可以得到希望的曲线图。

16.5.2　axes 函数

函数 axes 可以用来生成坐标轴并指定坐标轴为确定位置，其调用格式为：

```
axes('position', [x,y,w,h]);
```

参数说明：x 和 y 是坐标轴左下角点相对于 Figure 窗口的横、纵坐标值(坐标值是相对于 Figure 窗口的空白区域而言的)。w 和 h 是坐标轴的宽和高。

比如，调用函数 axes 绘制坐标轴的程序如下：

```
figure;                                              % 生成新的图形窗口
axes('Position',[0.11,0.13,0.4,0.3]);plot(rand(1,12));    % 生成坐标轴并绘图
axes('Position',[0.51,0.53,0.4,0.3]);plot(rand(1,12));    % 生成坐标轴并绘图
```

输出图形如图 16.43 所示，可见函数 axes 可以把坐标轴放置在图形窗口的不同位置上。

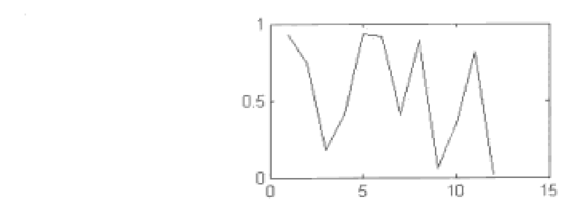
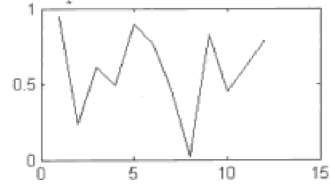



图 16.43　利用函数 axes 布局多个坐标轴窗口

16.5.3 图上图

有时为了在一个较大坐标轴上绘制一个一个小图来说明局部方法的结果，可以调用 axes 函数实现这个要求。下面通过绘制 $y=1/(t-3)$ 的曲线举例说明该函数的做法。程序如下：

```
figure;                                        % 生成新的图形窗口
t=linspace(0,6,300);t1=linspace(2.8,3.2,300);  % 生成曲线的整体离散坐标值 t 和局部离散坐标值 t1
y=sin(1./[t-3]);y1=sin(1./[t1-3]);             % 生成曲线的整体离散函数值 y 和局部离散函数值 y1
plot(t,y);axis('equal');                       % 绘制整体曲线图
axes('Position',[0.18,0.62,0.28,0.25]);        % 生成子图
plot(t1,y1);                                   % 绘制局部曲线图
xlim([min(t1),max(t1)]);                       % 设置坐标轴范围
```

输出结果如图 16.44 所示，图中给出了对其中变化剧烈的部分的放大处理，从而得到了更清楚的显示效果。

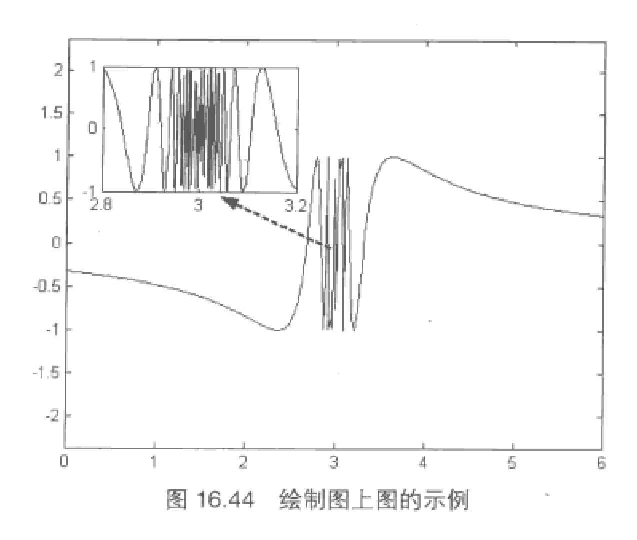


图 16.44 绘制图上图的示例

16.6 图像处理函数

本节介绍利用 MATLAB 进行基本图像处理的知识，包括图像的读写、剪切、旋转等。

一幅图像对应一个 $M\times N$（灰度图像）或者 $M\times N\times 3$（彩色图像，红 R、绿 G、蓝 B 三基色分量值）的数组。在 MATLAB 中提供了函数 imread 和 imwrite 来读写数字图像，它们的调用格式为：

```
a = imread('filename');
imwrite(A,'filename');
```

参数说明： A 是数字图像对应的数组。filename 是数字图像的文件名。MATLAB 支持的图片格式有 jpg，bmp，tiff，png 等。

在 MATLAB 显示图像的函数有 imshow 和 image，它们的调用格式为：

```
imshow(I,N)
image(I)
```

参数说明： I 是数字图像对应的数组。N 表示灰度级，其默认值是 256。

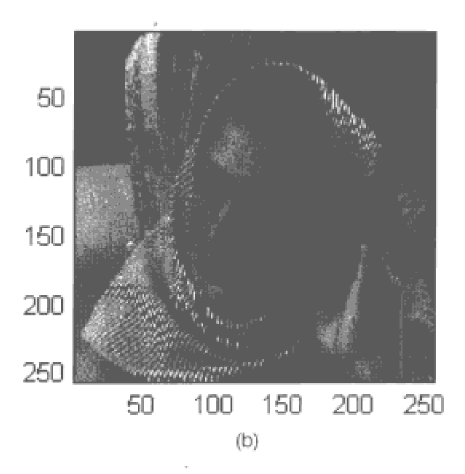
下面给出利用函数 imshow 和 image 显示图像的例子，程序如下：

```
figure;                                    % 生成新的图形窗口
C=imread('cameraman.tif');                 % 读入图像
load woman;                                % 导入图像数据（该图像是小波工具箱中带有的）
subplot(121);imshow(C);xlabel('(a)');      % 显示图像
subplot(122);image(X);xlabel('(b)');       % 显示图像
axis('square');                            % 设置坐标轴为方形
```

所得图形如图 16.45 所示，从中可以看出利用函数 imshow 显示的图片不带刻度值，而函数 image 显示的图片带有刻度。



(a)



(b)

图 16.45　图像显示



函数 imshow 显示图像时，坐标轴直接按图像的行数和列数对应比例显示，同时坐标轴自动设置为关闭状态。函数 image 显示图像时坐标轴的长宽比例不变，同时坐标轴处于显示状态。如果在调用函数 image 显示图像后，利用语句 "axis('image','off');" 设置坐标轴，则得到显示图像的效果和函数 imshow 一样。

函数 imshow 和 image 都是按正常方向显示图像，利用函数 surface 可以在图形窗口中把图像倾斜地显示，调用格式如下：

```
surface(X,Y,Z,C,'ProperttyName1','PropertyValue1',...);
```

参数说明： X，Y，Z 是坐标位置。C 是灰度数据。ProperttyName1 和 PropertyValue1 等是属性名称和相应的属性值。

例 16-17： 利用函数 surface 显示图像。

```
load woman;                                % 导入图像数据
A=imread('cameraman.tif');A=double(A);     % 读入数字图像
B=imread('rice.png');B=double(B);          % 读入数字图像
Z=zeros(size(X));Y=ones(size(X));          % 生成全 1 和全 0 矩阵
```

```
[xx,yy]=meshgrid(linspace(0,1,256));          % 生成网格坐标矩阵
surface(Y,xx,yy,flipud(X),'FaceColor','texturemap','EdgeColor','none','CDataMa
pping','direct');
colormap(map);view(3);xlim([0,1]);hold on;  % 设置颜色和坐标轴范围
surface(xx,Y,yy,flipud(A),'FaceColor','texturemap','EdgeColor','none','CDataMa
pping','direct');
surface(xx,yy,Z,flipud(B),'FaceColor','texturemap','EdgeColor','none','CDataMa
pping','direct'); %显示图像
view([-71,22]);                               % 设置视角
```

这里首先读入 3 幅图像，然后利用函数 surface 把它们分别放置在 XOY，XOZ 和 YOZ 平面上。输出图像如图 16.46 所示。

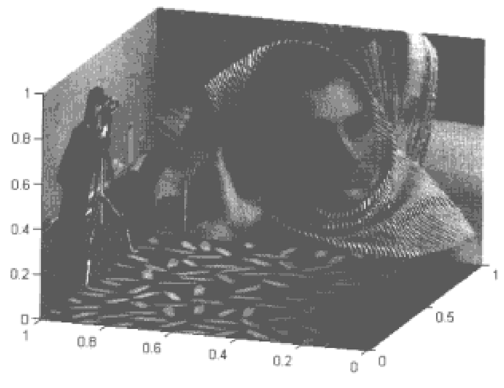


图 16.46 倾斜显示图像



可以通过改变函数 view 的参数来旋转图像的角度，从而得到任意角度显示。

通过提取数组中不同位置的数据，可以实现图像剪切的目的。另外，MATLAB 还提供了一个专门的图像剪切函数，即 imcrop，其调用格式为：

```
I2= imcrop(I, rect);
I2= imcrop(I,map,rect);
```

参数说明： I 是图像数据矩阵。rect 用于定义剪切的区域，其由 4 个元素组成，即[x,y,w,h]，其中 x 和 y 表示切下的子图像在原图像右上角点的坐标，w 和 h 分别表示切下子图像的宽和高。

下面举例说明剪切图像。

```
figure;              % 生成新的图形窗口
load woman;          % 导入图像数据
subplot(131);imshow(X,[]);  % 显示图像
xlabel('(a)','Fontsize',12);    % X 轴标注
Xc=ones(size(X))*255;        %生成空白图像
Xc(65:192,65:192)=X(65:192,65:192);   % 切出中心区域的图像数据
subplot(132);imshow(Xc,[]);            % 显示图像
xlabel('(b)','Fontsize',12);              % X 轴标注
Xs= imcrop(X,map,[1,1,128,64]);        % 剪切矩阵数据
subplot(133);imshow(Xs,[]);            % 显示图像
xlabel('(c)','Fontsize',12);              % X 轴标注
```

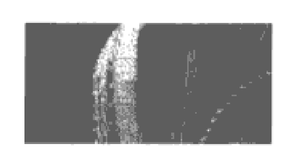
这里读入 woman.mat 数据对应的图像，然后通过提取矩阵中部分元素实现图像剪切的目的，最后给出利用函数 imcrop 实现剪切的任务。原图和剪切后的图像利用函数 imshow 显示出来。输出图形如图 16.47 所示，其中图(b)和(c)给出了原图部分数据的显示。



(a)



(b)



(c)

图 16.47　图像剪切：(a) 原图，(b) 切出的中心区域

用函数 imrotate 可以旋转图像，其调用格式为：

```
B = imrotate(A,angle, 'method', 'bbox');
```

参数说明：B 是旋转后图像的数据。A 是输入图像。angle 表示旋转的角度。参数 method 用于指定计算过程中采用的插值方法：nearest（其为默认值）表示最近邻插值，bilinear 表示双线性插值，bicubic 表示双三次插值。参数 bbox 用于控制旋转后图像大小是否改变，可选参数为：crop 表示剪切旋转后的图像中间部分返回和原图像大小一样的图像，loose（其为默认值）表示缩小图像，图像的边缘还在返回的图像中。

下面调用函数 imrotate 来旋转图像，相应程序如下：

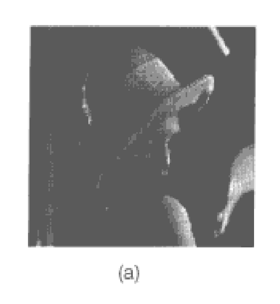
```
A = imread('lena.bmp');                    % 读入原图像
R1 = imrotate(A,-10,'bilinear','crop');        % 将图像顺时针旋转 10°
R2 = imrotate(A,-10,'bilinear');            % 将图像顺时针旋转 10°
subplot(131);imshow(A);    % 显示原始图像
xlabel('(a)','Fontsize',12);    % X 轴标注
subplot(132);imshow(R1);    % 显示旋转后的图像
xlabel('(b)','Fontsize',12);    % X 轴标注
subplot(133);imshow(R2);    % 显示旋转后的图像
xlabel('(c)','Fontsize',12);    % X 轴标注
```

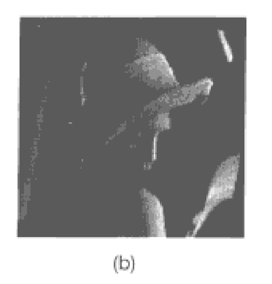
这里给出了利用函数 imrotate 把 lena 图像顺时针旋转 10° 的例子，其中分别给出保持图像大小不变和缩小一定比例的两种结果。输出图形如图 16.48 所示，其中图(b)图像大小不变，图(c)缩小一定比例使图像内容不被切掉。

函数 imresize 可以改变图像的像素数目，该函数调用格式为：

```
B = imresize(A,m,'method');
```

参数说明：B 是缩放后的图像数据。A 是输入的原始图像。m 是缩放比例。method 用于指定缩放计算中的插值方法：nearest（其为默认值）表示最近邻插值，bilinear 表示双线性插值，bicubic 表示双三次插值。


(a)


(b)

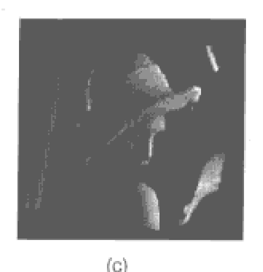

(c)

图 16.48 图像旋转

下面举例说明函数 imresize 的用法。

```
A = imread('bb.bmp');          % 读入图像
Z1 = imresize(A,0.5,'bilinear');  % 将图像缩小到 0.75 倍
Z2 = imresize(A,2,'bicubic');    % 将图像放大到 1.25 倍
s(1)=subplot(131);
image(A);    % 显示原始图像
xlabel('(a)','Fontsize',12);       % X 轴标注
s(2)=subplot(132);
image(Z1);                    % 显示缩小后的图像
xlabel('(b)','Fontsize',12);       % X 轴标注
s(3)=subplot(133);
image(Z2);                    % 显示放大后的图像
xlabel('(c)','Fontsize',12);       % X 轴标注
colormap(gray);axis(s,'image');  % 显示为灰度图,设置坐标轴属性
```

上述程序给出了利用函数 imresize 改变矩阵的大小，包括缩小和放大两种情况。输出图形如图 16.49 所示，从刻度可以看出图(b)数据量缩小而图(c)数据量增加。

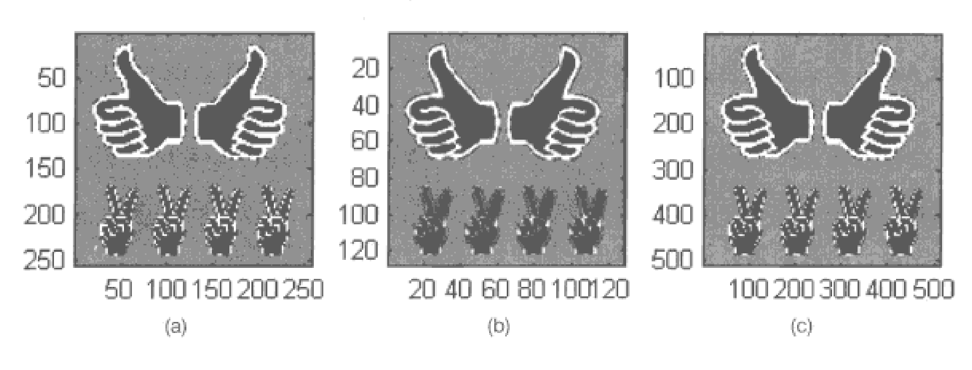


图 16.49 图像缩放

从图 16.49 中的坐标轴数值可以读出图像像素的数目。更多图像处理函数可以在 MATLAB 的安装路径下查询，即\MathTools\MATLAB\ R2008a\toolbox\images\images。

16.7 动画的绘制

利用动画可以观察变化全过程的更多信息，同时可能得到相关的规律。本节介绍如何利用 MATLAB 制作动画，同时给出相关的例子。

16.7.1 制作动画的方法

在 MATLAB 中可以通过动画来显示一个变化的过程，动画的制作可以有下面两种方式：

- 在图形窗口中不断地计算并画出图形，在当前图形窗口中只显示利用当前参数计算所得的图片，这样看起来就具有动画效果了。这种方式适用于获得图像时间较短的情况。
- 在保存多幅图像后以电影的形式播放出来，这种方式适用于获取一幅图像时间较长的情况，用户可以在计算出所有图像后反复观察图像变化过程。

对于变化曲线的数据，可以通过改变句柄中的属性“XData”、“YData”和“ZData”的值来改变曲线的形状。

16.7.2 保存动画

记录动画可以用函数 getframe 来记录图形窗口的内容，该函数的调用格式为：

```
f=getframe(H);
```

参数说明： f 是记录的结果。H 是图形窗口或者坐标轴对应的句柄。下面是调用函数 getframe 的基本结构：

```
for k=1:n
   plot_command;             % 绘图程序段
   M(k) = getframe(gcf);          % 记录当前窗口内容到 M 中
end
movie(M);                   % 以电影的方式播放记录的内容
```

上面介绍的 getframe 函数保存内容到 MATLAB 的变量空间（workspace）中，此外 MATLAB 还提供函数 avifile 来记录动画过程，利用这个函数可以得到一个 avi 文件。该函数调用格式为：

```
aviobj = avifile('filename');
aviobj = avifile('filename','propertyname',value,...);
```

参数说明： aviobj 用于标记 avi 文件，相当于一个句柄。filename 是 avi 文件的名称。propertyname 和 value 分别是该函数的属性和相应取值，用于控制 avi 文件的质量等属性。

例 16-18： 函数 avifile 的例子。

```
fig=figure;
set(fig,'DoubleBuffer','on');            % 设置渲染效果
set(gca,'xlim',[-80 80],'ylim',[-80 80],...
   'NextPlot','replace','Visible','off')   % 设置坐标轴范围、绘图方式和不可见
mov = avifile('example.avi')           % 定义 avi 文件名
x = -pi:.1:pi;                       % 生成离散数据
radius = [0:length(x)];                % 生成离散数据
for i=1:length(x)
   h = patch(sin(x)*radius(i),cos(x)*radius(i),[abs(cos(x(i))) 0 0]);  % 填充区域
   set(h,'EraseMode','xor');           % 设置擦除方式
   F = getframe(gca);               % 获取当前坐标轴图像信息
   mov = addframe(mov,F);           % 把当前图像添加到 avi 文件中
end
mov = close(mov);                   % 关闭句柄 mov
```



上面对相关语句作用做了注释。在利用 avifile 函数制作 avi 文件过程中记录的图形窗口区域不能被其他界面遮挡，否则记录区域的内容被挡住。

16.7.3 实例

本小节举例说明这两种方式制作动画的过程。首先绘制动态的 Koch 曲线。该曲线的定义如图 16.50 所示。

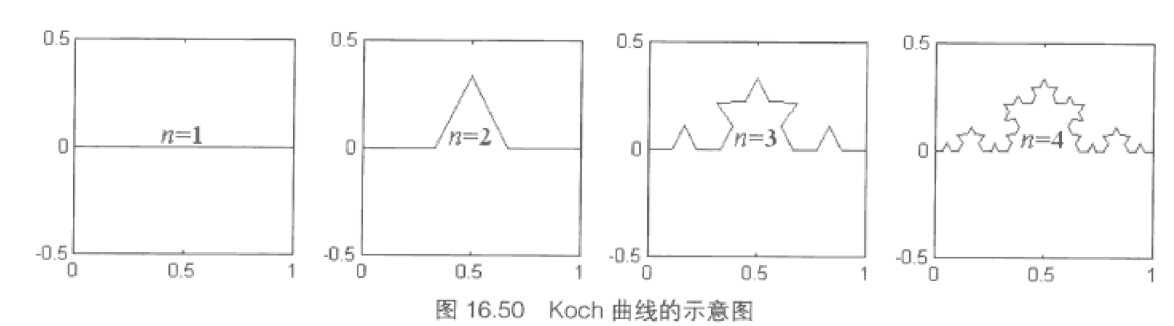


图 16.50 Koch 曲线的示意图

可见 Koch 曲线是三等分线段，中点处的分点向外移动到 $\sqrt{3}/2$ 倍线段长的位置处，依次连接起来就得到 Koch 曲线。不断地按照这个规律就得到不同 n 对应的曲线，这种曲线即著名的分形曲线之一。这里考虑把中点向外移动的过程做成动画。

首先定义函数 kochxy 计算 Koch 曲线的三等分点、中点和中点处法线的角度，这样的函数定义如下：

```
function [z1,A]=kochxy(z);
z1=z(1);A=[];                          % 初始化坐标 z1 和角度 A
for k=1:length(z)-1;
  A=[A,angle(z(k+1)-z(k))+pi/2];      % 记录线段中点的法线方向
  z1=[z1,[2*z(k)+z(k+1)]/3,[z(k)+z(k+1)]/2,[z(k)+z(k+1)*2]/3,z(k+1)];  % 计算三等分点、中点并记录
end
```

参数说明： z1（其为一个复数）表示下一级 Koch 曲线端点、三等分点和中点向量。A 是中点处法线的角度。z 是当前一级 Koch 曲线端点和折点的坐标值（其为一个复数）。

调用函数 kochxy 计算关键点，用更新曲线数据的方法制作 Koch 曲线动态过程的程序如下：

```
figure;                                    % 生成新的图形窗口
set(gcf,'Color','w','DoubleBuffer','on');  % 设置图形窗口的位置、大小、背景色和渲染效果
z=[0,1];      % 位置初值
N=8;        % 线段中点升高的次数
hp=plot(real(z),imag(z),'r');      % 绘制初值对应的曲线
axis([0,1,0,1],'square');         % 设置坐标轴范围和宽高比例
for n=1:4;
    [z,A]=kochxy(z);          % 得到 Koch 曲线的分点数据
    d=abs(z(2)-z(1));         % 计算子线段长度
    dd=d/N;                 % 中点每次升高的步长
    for k=1:N;
        z(3:4:end)=z(3:4:end)+dd*exp(i*A);      % 更新中点坐标
        set(hp,'XData',real(z),'YData',imag(z));  % 更新曲线数据
        pause(0.2);                           % 停顿一下
```

```
    end
end
```

其中利用函数 pause 进行 0.2 秒的停顿突出变化效果，最终生成图案如图 16.51 所示，中间的动画过程用户可以运行上面的程序。

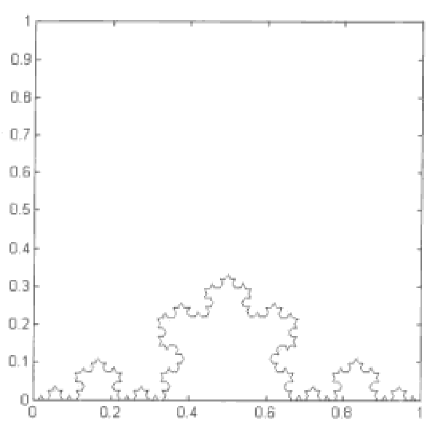


图 16.51　动画最终生成的图案

第 2 个示例是螺旋片绕其轴线旋转的动画。光盘中\Ch16 文件夹下的 helix.m 文件是一个绘制螺旋片的函数文件（该程序较长，这里就不显示了，感兴趣的用户可以自己研究这个程序），其调用格式为：

```
helix(A)
```

参数说明： A 是一个角度参数，用于确定螺旋片的旋转角度。调用这个函数可以在坐标轴上绘制一个螺旋片。

调用上面的函数 helix 来绘制一个旋转动画，相关语句如下：

```
figure;                         % 生成新的图形窗口
set(gcf,'Position',[11 153 995 420],'Color','w','DoubleBuffer','on');  % 设置图形
窗口的位置等
A=linspace(0,pi*4,40);          % 生成角度变化序列
subplot(121);helix(0);          % 在第一子图中绘制特定角度的螺旋片
subplot(122);
for k=1:length(A);
   helix(A(k));                 % 绘制不同角度的螺旋片
   text(0,3,3,['Time = ',num2str(k)],'Fontsize',18,'Fontname','Times new roman');
% 显示变化的参数
    M(k)=getframe(gca);         % 记录到动画的帧序列中
end
figure;                         % 生成新的图形窗口
axis off;                       % 关闭轴标轴
movie(M,1)                      % 播放一次记录的动画
```

首先在图形窗口左侧画出一个静止的螺旋片，然后调用函数 helix 在右侧的坐标轴窗口内画出不同旋转角度的螺旋片。最后利用函数 movie 播放动画。记录过程中的一帧如图 16.52 所示，利用函数 movie 播放动画的界面如图 16.53 所示。

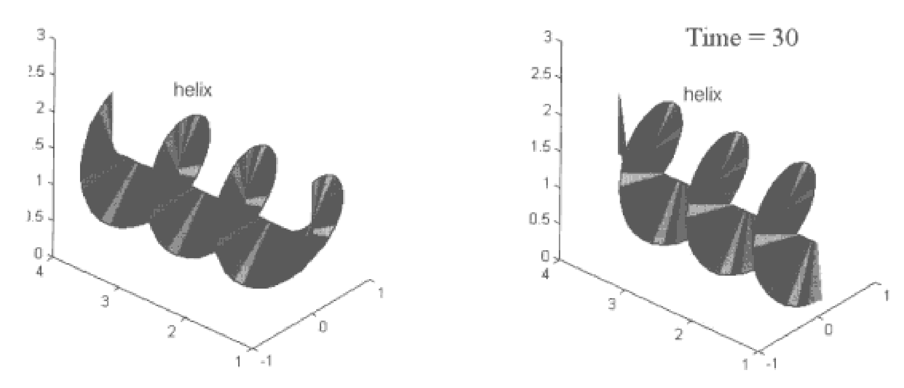


图 16.52 记录旋转过程中的一帧

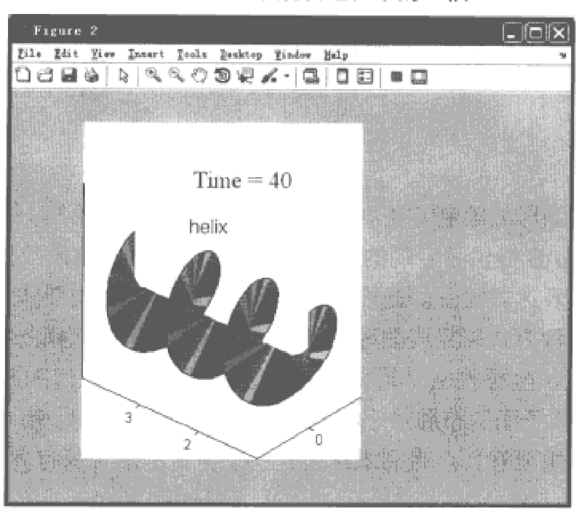


图 16.53 播放动画的界面

16.8 图形的保存

如果利用语句来实现计算所得图形的保存，在不断修改程序中的参数时就显得很方便。保存图形的常用函数有 saveas 和 print。saveas 的调用格式如下：

```
saveas(h,'filename')
```

参数说明： h 是图形窗口或者坐标轴的句柄。filename 是图像文件的名称，图片文件的扩展名可以是 bmp，jpg，tiff，png 等，其中 tiff 和 png 是科技写作中常用的图片格式。比如下面的语句可以把图形窗口的内容保存到图片文件中：

```
saveas(gcf,'abc.tiff')
saveas(gcf,'cde.png')
```



调用函数 saveas 保存图像时，需要把该函数放在图形已经绘制完毕的位置。

利用函数 print 可以像打印文档一样打印图形窗口中的内容，该函数的调用格式为：

```
print -options filename
```

参数说明：options 用于指定打印的选项，如图片类型、dpi 等。filename 是文件名（不包括扩展名）。

下面给出 jpg，tiff，png，eps 和 ps 文件的语法格式：

```
print -djpeg90 -r300 Liu   % 打印为 jpg 文件
print -dtiff -r300 Liu      % 打印为 tiff 文件
print -dtiffnocompression -r300 Liu   % 打印为无压缩 tiff 文件
print -dpng -r300 Liu       % 打印为 png 文件
print -depsc -tiff -r300 Liu    % 打印为 eps 文件
print -dpsc -r300 Liu       % 打印为 ps 文件
```

其中 r300 用于定义 dpi 的数值（300 为 dpi 的数值），用户可以根据需要选用不同的数值。eps 图形文件一般用于编辑软件 CTeX 的文档中。

16.9 小结

本章主要介绍了二维图形的绘制方法。首先介绍了二维曲线的绘制，给出了常用函数的调用格式。对于特殊图形，MATLAB 提供了专门的函数来实现。除了利用离散数据绘图外，用户还可以利用符号绘图函数根据数学表达式绘图。图形编辑对于美化图形具有重要作用，MATLAB 提供了交互式编辑窗口和函数控制两种方式来完成图形的编辑任务，其中掌握句柄的控制是核心部分。利用基本函数可以实现坐标刻度、坐标网格、箭头等的重绘。根据基本几何图形的关键点可以利用函数 plot 来绘制常见的图形。多子图的布局对于管理较多曲线有重要作用。基本图像处理操作和动画制作对于视觉上观察结果有重要意义。最后本章还介绍了常见的图片保存方法。

◆ ◆ ◆

第 17 章　三维数据可视化

本章包括

- ◆ **基本函数**　介绍基本三维绘图函数的用法及曲面的剪裁知识。
- ◆ **彩色图及颜色条**　介绍彩色图形的颜色管理以及色轴的添加方法。
- ◆ **无色网格曲面**　介绍两种单色网格曲面的绘制方法。
- ◆ **视角与光照**　介绍控制视角和光照效果的函数及相关方法。
- ◆ **图形的注释**　介绍三维图形的注释方法以及带有斜体希腊字母图形的输出方法。

前一章主要介绍了二维绘图的知识。与二维图形相比，三维图形的信息量远远大于二维图形，而且对于统计函数与不同变量之间的关系具有重要的辅助作用。本章将介绍 MATLAB 在三维绘图方面的知识，如三维曲线、曲面、不规则图形的绘制等方面的编程要点。

17.1 基本函数

本节主要介绍在 MATLAB 中三维图形的基本函数，包括它们的用法和相应的举例说明。利用它们可以得到不同样式的三维图形。

17.1.1 函数 meshgrid

进行三维绘图的一个常用函数是 meshgrid，其调用格式为：

```
[X,Y] = meshgrid(x)
[X,Y] = meshgrid(x,y)
```

参数说明： X 和 Y 表示网格点的坐标。x 和 y 分别表示 X 轴和 Y 轴方向的离散采样点，当 y 默认时选择 y=x。函数 ndgrid 可以生成 N 维网格坐标，其用法和 meshgrid 函数相似。

下面举例说明函数 meshgrid 的使用。

```
[X,Y]=meshgrid(1:4,5:8)
```

输出结果为：

```
X =      1     2     3     4
        1     2     3     4
        1     2     3     4
     1     •2     3     4
Y =      5     5     5     5
        6     6     6     6
        7     7     7     7
        8     8     8     8
```

可见 X 和 Y 分别是水平和竖直方向变化的坐标。数据 X 和 Y 用于在三维绘图函数中表示坐标。

MATLAB 提供的三维绘图函数如表 17.1 所示。

表 17.1　三维绘图函数

函数名	说明	函数名	说明
line，plot3，ezplot3	绘制三维曲线	surfl	绘制带有光照的三维曲面图
mesh，ezmesh	绘制三维网状图	surfnorm	计算或者显示三维表面法向
meshc，ezmeshc	绘制带有等高线的三维网状图	contour3	绘制三维等高线图
meshz	绘制带有"围裙"的网状图	waterfall	绘制带有水流效果的三维图
surf，ezsurf	绘制三维曲面图	pcolor	绘制以颜色表示高度的图形
surfc，ezsurfc	绘制带有等高线的三维曲面图		

下面依次介绍这些函数的用法。

17.1.2　三维曲线

函数 line 绘制三维曲线的语法格式为：

```
line(x,y,z)
```

参数说明： x，y 和 z 表示三维曲线在相应轴的数据。

函数 plot3 可用来绘制三维曲线，其调用格式为：

```
plot3(x,y,z)
plot3(x,y,z, 's')
plot3(x1,y1,z1, 's1',x2,y2,z2, 's2',...)
plot3(...,'Property1',PropertyValue1,...)
```

参数说明： x，y 和 z 表示三维曲线在相应轴的数据。s，s1 和 s2 等用于指定曲线的线型、颜色和标记符号等信息，它们的取值可参见表 16.2。x1，y1，z1，x2，y2 和 z2 等表示三维曲线在相应轴的数据，用于同时绘制多条三维曲线。Property1 和 PropertyValue1 是曲线属性和相应取值。

函数 ezplot3 用于根据符号表达式绘制函数曲线，其调用格式为：

```
ezplot3(xfun,yfun,zfun)
ezplot3(xfun,yfun,zfun,[tmin,tmax])
```

参数说明： xfun，yfun 和 zfun 是 3 个一元函数。tmin 和 tmax 分别表示自变量的下限和上限。

下面举例说明函数 line，plot3，ezplot3 的用法，程序如下：

```
figure;                                          %生成新的图形窗口
t=linspace(0,pi,401);                                  % 生成离散采样点
xf=inline('sin(t*8)*2');yf=inline('cos(t*8)*3');    % 定义内联函数
s(1)=subplot(131);line(sin(t*8),cos(t*8),t);       % 利用函数 line 绘制三维曲线
s(2)=subplot(132);plot3(sin(t*8)/2,cos(t*8)/2,t,'k',sin(t*16),cos(t*16),t,'r:'
);  % plot3 绘制两条三维曲线
s(3)=subplot(133);ezplot3(xf,yf,inline('t'),[-3,3]);axis equal;  % 根据符号表达式
绘制三维曲线
view(s(1),[-33,14]);view(s(2),[-33,14]);view(s(3),[44,62]);   % 设置坐标轴的视角
```

这里给出利用等距螺旋线的绘制，inline 函数定义了含参数 t 的函数形式。分别利用函数 line，plot3 和 ezplot3 进行三维曲线的绘制。

运行上述程序，所得曲线如图 17.1 所示。

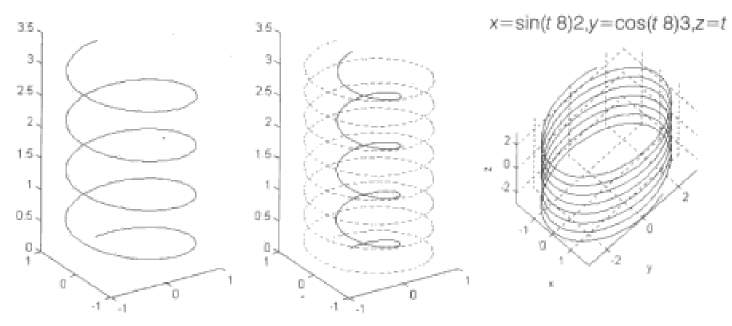


图 17.1　三维曲线的绘制



利用函数 ezplot3 绘制三维曲线时，MATLAB 自动进行 X 轴、Y 轴、Z 轴和图题的标注。坐标轴的视角可以通过函数 view 来调整。

17.1.3　三维网格图

函数 mesh 可以绘制三维网格图，其调用格式为：

```
mesh(Z);
mesh(X,Y,Z);
mesh(X,Y,Z,C);
mesh(...,'PropertyName',PropertyValue,...);
```

参数说明： X，Y，Z 分别表示网状图节点在 X 轴、Y 轴和 Z 轴的坐标。X 和 Y 是向量或者矩阵，Z 是向量，其中 X 和 Y 默认为 1:n 和 1:m（m 和 n 对应于矩阵 Z 的列数和行数）。C 表示网格线段的颜色，其默认值为 Z。PropertyName 和 PropertyValue 表示函数 mesh 的属性和相应取值。

下面给出利用函数 mesh 绘制不同着色方式的网状图，实现程序如下：

```
figure;                              %生成新的图形窗口
[X,Y] = meshgrid(-2:.2:2, -2:.2:2);   % 生成坐标网格矩阵
Z = X .* exp(-X.^2 - Y.^2);           % 计算二元函数 z(x,y)的离散数值
subplot(131);mesh(X,Y,Z);            % 绘制网状图
xlabel('(a)','Fontsize',14,'Fontname','Times New Roman');       % X 轴标柱
subplot(132);mesh(X,Y,Z,rand(size(Z)));                 % 绘制随机彩色网状图
xlabel('(b)','Fontsize',14,'Fontname','Times New Roman');       % X 轴标柱
subplot(133);m=mesh(X,Y,Z,2*ones(size(Z)),'EdgeColor','k');   % 绘制单色网状图
xlabel('(c)','Fontsize',14,'Fontname','Times New Roman');       % X 轴标柱
```

这里利用 mesh 函数对二元函数绘制相应的网格图，其中给出 3 种不同着色方式。运行上述程序，所得的图形如图 17.2 所示。

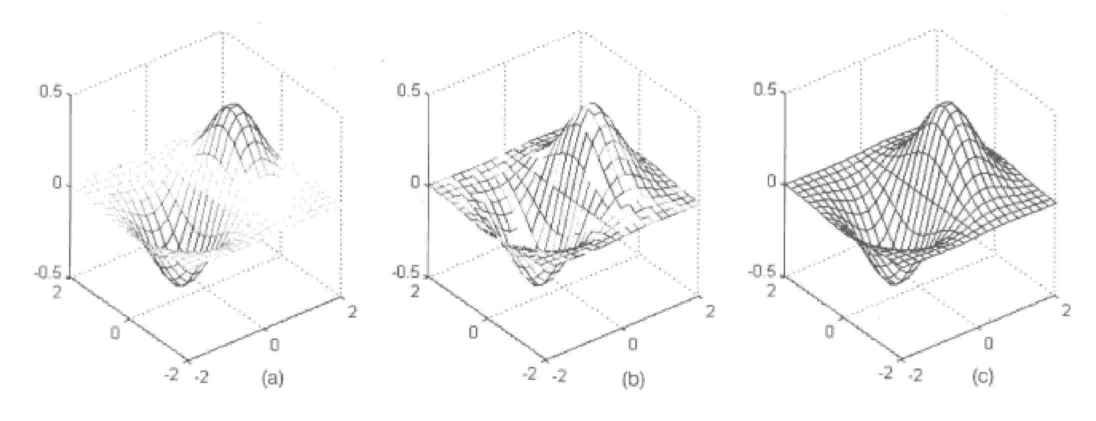


图 17.2 带有不同着色方式的网状图



在属性 EdgeColor 中设置不同颜色可以进行单一着色的网格（如图 17.2(c)所示），其中颜色可以使用特殊颜色控制符，也可以利用一个含 3 个元素的向量定义。

17.1.4 用 ezmesh 绘制三维网格图

函数 ezmesh 可以根据数学表达式绘制三维网格图，其调用格式有：

```
ezmesh(zfun);
ezmesh(zfun,domain);
ezmesh(xfun,yfun,zfun,[xmin,xmax,ymin,ymax]);
ezmesh(...,N);
```

参数说明：xfun，yfun和zfun表示相应坐标轴的数学表达式，其可以利用函数inline定义。domain表示数学表达式中变量的取值范围，它是含 2 个或者 4 个元素的向量，即[a, b]或者[a, b]，[c, d]，前面给出的区间是限定变量范围的，domain 的默认值为[-2*pi,2*pi,-2*pi,2*pi]。向量[smin,smax,tmin,tmax]用于限制 xfun，yfun 和 zfun 中变量的范围。N 用于指定变量的采样点数。

例 17-1：利用函数 ezmesh 绘图。

```
figure;                          %生成新的图形窗口
zfun=inline('sqrt(1-s^2-t^2)');    % 根据表达式定义内联函数
xs=inline('cos(s)*cos(t)');       % 根据 xfun 的表达式
ys=inline('cos(s)*sin(t)');       % 根据 yfun 的表达式
zs=inline('sin(s)');             % 根据 zfun 的表达式
subplot(131);ezmesh(zfun,100);    % 绘制网状图并指明采样点数
xlabel('(a)','Fontsize',14,'Fontname','Times New Roman');  % X 轴标柱
subplot(132);ezmesh(zfun,[-1,1],20);                      % 绘制网状图并指明采样点数
xlabel('(b)','Fontsize',14,'Fontname','Times New Roman');  % X 轴标柱
subplot(133);ezmesh(xs,ys,zs,[-pi,pi],20);                   % 根据 xfun，yfun 和 zfun 绘
制网状图
xlabel('(c)','Fontsize',14,'Fontname','Times New Roman');   % X 轴标柱
```

利用函数 inline 定义球坐标系下的球面，通过参数 s 和 t 表示方位角。通过设置参数范围给出半球和完整球面图形。运行上述程序，得到如图 17.3 所示的图形。

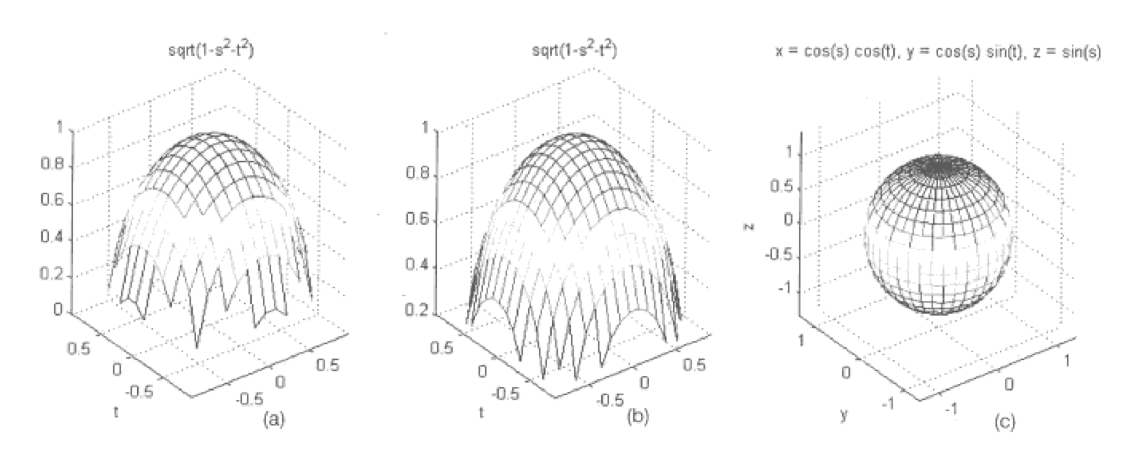


图 17.3 利用函数 ezmesh 绘制网状图

在调用函数 ezmesh 绘图时，MATLAB 自动添加坐标轴标注和图题。在图 17.3(c)中，结果是一个圆球，其中 xfun，yfun 和 zfun 的表达式是根据圆球的极坐标表达式确定的。函数 ezmesh 不支持通过属性方式改变其着色方案，同时也不支持句柄输出。为了实现对函数 ezmesh 绘制图形的着色方案进行调整，需要通过坐标轴句柄来查找 ezmesh 绘制的图形对象，相关语句为：

```
S=get(gca,'Children');
```

其中 Children 是当前坐标轴句柄下面的子句柄。S 对应着网状图的句柄，通过 S 可以对网状图的属性进行编辑，从而实现对网状图着色方案的编辑。

下面以上半球的网状图为例给出着色方案编辑的代码。

```
figure;                          %生成新的图形窗口
xs=inline('cos(s)*cos(t)');   % 根据 xfun 的表达式
ys=inline('cos(s)*sin(t)');   % 根据 yfun 的表达式
zs=inline('sin(s)');          % 根据 zfun 的表达式
subplot(131);ezmesh(xs,ys,zs,[0,pi],16);view([-39,56])     % 根据 xfun,yfun 和 zfun
绘制网状图
xlabel('(a)','Fontsize',14,'Fontname','Times New Roman');  % X 轴标柱
subplot(132);ezmesh(xs,ys,zs,[0,pi],16);view([-39,56])     % 根据 xfun,yfun 和 zfun
绘制网状图
S1=get(gca,'Children');set(S1,'CData',rand(size(get(S1,'CData'))));   % 通过句柄
对网状图着色
xlabel('(b)','Fontsize',14,'Fontname','Times New Roman');   % X 轴标柱
subplot(133);ezmesh(xs,ys,zs,[0,pi],16);view([-39,56])      % 根据 xfun, yfun 和
zfun 绘制网状图
S2=get(gca,'Children');set(S2,'EdgeColor','k');             % 通过句柄对网状图着色
xlabel('(c)','Fontsize',14,'Fontname','Times New Roman');   % X 轴标柱
```

这里改变颜色通过句柄 gca，查找下面的子句柄而得到曲面对象对应的句柄。对其中的 CData 属性值进行修改可以得到更改着色的目的。运行上述程序，输出如图 17.4 所示。

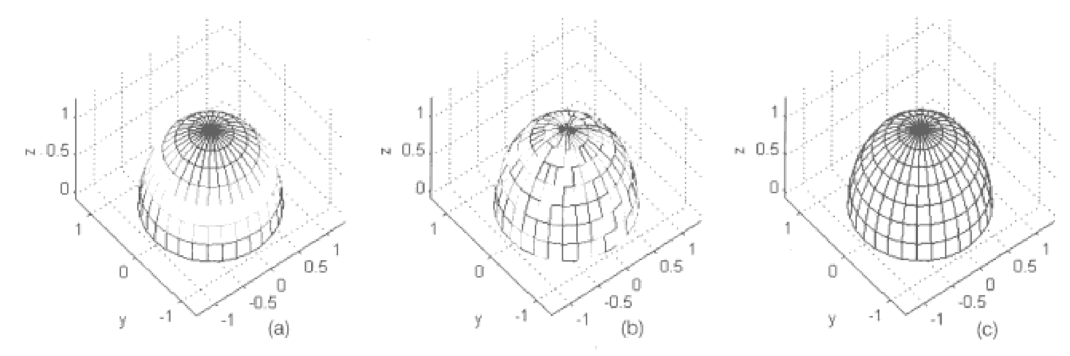


图 17.4 改变函数 ezmesh 绘制的网状图的着色方案



在图 17.4 中，图(a)是默认的着色方案，图(b)是随机着色的结果，图(c)是单色的。

17.1.5 带有等高线的网状图

函数 meshc 可绘制带有等高线的网状图，其调用格式与函数 mesh 相似，其调用格式为：

```
meshc(Z)
meshc(X,Y,Z)
meshc(X,Y,Z,C)
```

参数说明： X，Y，Z 分别表示网状图节点的在 X 轴、Y 轴和 Z 轴的坐标，X 和 Y 是向量或者矩阵，Z 是向量，其中 X 和 Y 默认为 1:n 和 1:m（m 和 n 对应于矩阵 Z 的列数和行数）。C 表示网格线段的颜色，其默认值为 Z。与 mesh 函数不同的是用户不能在函数 meshc 的输入参数中通过属性改变网格曲线或者等高线的属性值，但是可以通过句柄来改变函数 meshc 绘制的网状图。

下面举例说明函数 meshc 的用法，程序代码如下：

```
figure;                        %生成新的图形窗口
[X,Y,Z]=peaks(30);              % 生成坐标数据
subplot(121);meshc(Z);    % 生成带等高线的网状图
xlabel('(a)','Fontsize',14,'Fontname','Times New Roman');  % X 轴标柱
subplot(122);m=meshc(X,Y,Z,Z);                   % 生成带等高线的网状图，并输出句柄
set(m(1),'EdgeColor','k');                         % 通过句柄改变属性值
xlabel('(b)','Fontsize',14,'Fontname','Times New Roman');   % X 轴标柱
```

运行上述程序，所得图形如图 17.5 所示。



代码中的输出句柄 m 是由多个元素组成的向量，其第一个元素对应着网状曲面的图形，而其他元素对应着等高线。这里通过设置 m(1)的 EdgeColor 属性把网状图的颜色改为黑色。

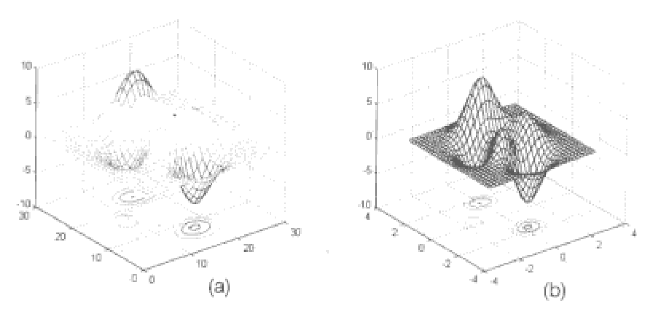


图 17.5 带等高线的网状图

17.1.6 带有等高线的网状图

函数 ezmeshc 可以根据数学表达式绘制带有等高线的网状图，其调用格式为：

```
ezmeshc(zfun)
ezmeshc(zfun,DOMAIN)
ezmeshc(xfun,yfun,zfun,[smin,smax,tmin,tmax])
ezmeshc(...,N)
ezmeshc(...,'circ')
```

参数说明：xfun，yfun和zfun表示相应坐标轴的数学表达式，其可以利用函数inline定义。domain表示数学表达式中变量的取值范围，它是含2个或者4个元素的向量，即[a, b]或者[a, b]，[c, d]，前面给出的区间是限定变量范围的，domain 的默认值为[-2*pi,2*pi,-2*pi,2*pi]。向量[smin,smax,tmin,tmax]用于限制 xfun，yfun 和 zfun 中变量的范围。N 用于指定变量的采样点数。circ 表示在 domain 参数限定区域内的一个圆形区域中绘制网状图。

下面调用函数 ezmeshc 绘制网状图，程序如下：

```
figure;                                                    %生成新的图形窗口
subplot(131);ezmeshc('imag(atan(x + i*y))',[-4,4]);
xlabel('(a)','Fontsize',14,'Fontname','Times New Roman');   % X轴标柱
subplot(132);ezmeshc('real(log(x + i*y))',[-4,4],30);
xlabel('(b)','Fontsize',14,'Fontname','Times New Roman');   % X轴标柱
S2=get(gca,'Children');set(S2(end),'EdgeColor','k');        % 通过句柄对网状图着色
subplot(133);ezmeshc('real(log(x + i*y))',[-4,4],'circ')
xlabel('(c)','Fontsize',14,'Fontname','Times New Roman');   % X轴标柱
```

运行上述程序，输出图形如图 17.6 所示。

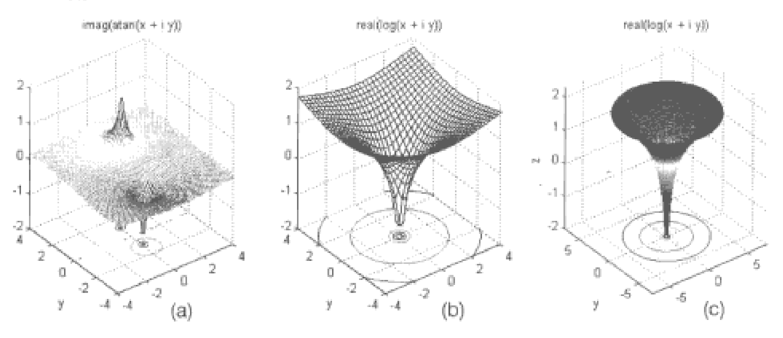


图 17.6 利用函数 ezmeshc 绘制的网状图



在获取函数 ezmeshc 所绘图形的句柄时，在句柄 S2 中，最后一个元素对应上面的网状曲面，而其他元素对应于下面的等高线。

17.1.7 带有“围裙”的网状图

函数 meshz 可以用来绘制带有“围裙”的网状图，其调用格式为：

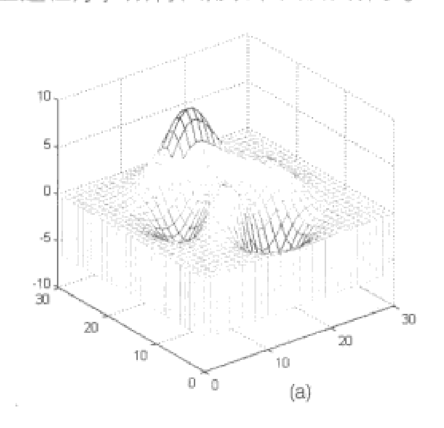
```
meshz(Z);
meshz(X,Y,Z);
meshz(X,Y,Z,C);
```

参数说明： X，Y，Z 分别表示网状图节点的在 X 轴、Y 轴和 Z 轴的坐标，X 和 Y 是向量或者矩阵，Z 是向量，其中 X 和 Y 的默认值为 X = 1:n 和 Y = 1:m（m 和 n 对应于矩阵 Z 的列数和行数）。C 表示网格线段的颜色，其默认值为 Z。与 mesh 函数不同的是用户不能在函数 meshz 的输入参数中通过属性改变网格曲线或者等高线的属性值，但是可以通过句柄来改变函数 meshz 绘制的网状图。

例 17–2： 调用函数 meshz 绘图。

```
figure;                          %生成新的图形窗口
[X,Y,Z]=peaks(30);               % 生成坐标数据
subplot(121);meshz(Z);           % 生成带“围裙”的网状图
xlabel('(a)','Fontsize',14,'Fontname','Times New Roman');  % X 轴标柱
subplot(122);m=meshz(X,Y,Z,Z);                     % 生成带“围裙”的网状图，并输出句柄
set(m,'EdgeColor','k');                              % 通过句柄改变网状图的着色方案
xlabel('(b)','Fontsize',14,'Fontname','Times New Roman');  % X 轴标柱
```

运行上述程序，所得图形如图 17.7 所示。



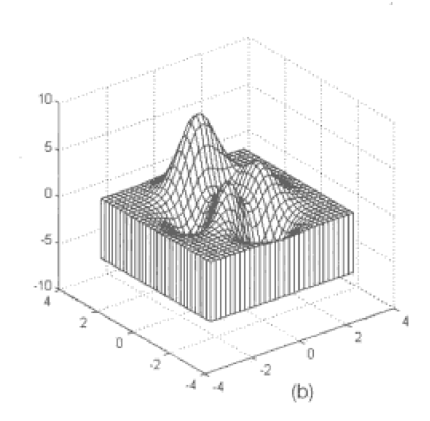


图 17.7 带“围裙”的网状图



函数 meshz 绘制图形的输出句柄 S2 是单个数值而非一个向量。如果用户利用函数 zlim 设置坐标轴范围将会使“围裙”部分变形，MATLAB 不能相应地切去多余的“围裙”（部分“围裙”溢出坐标轴范围）。

17.1.8 三维曲面图

函数 surf 可以绘制三维曲面图，其调用格式为：

```
surf(Z)
surf(Z,C)
surf(X,Y,Z,C)
surf(...,'PropertyName',PropertyValue,...)
```

参数说明：X，Y，Z 分别表示网状图节点的在 X 轴、Y 轴和 Z 轴的坐标，X 和 Y 是向量或者矩阵，Z 是向量，其中 X 和 Y 的默认值为 X= 1:n 和 Y= 1:m（m 和 n 对应于矩阵 Z 的列数和行数）。C 表示网格线段的颜色，其默认值为 Z。PropertyName 和 PropertyValue 表示函数 mesh 的属性和相应取值。

下面给出调用函数 surf 绘图的例子，程序如下：

```
figure;                                      %生成新的图形窗口
[x,y]=meshgrid(linspace(-3,3,21));  % 生成网格坐标
z=(x-y).*exp(-(x.^2+y.^2)/2);           % 生成离散的二元函数值
subplot(121);surf(x,y,z);              % 绘制曲面图
xlabel('(a)','Fontsize',14,'Fontname','Times New Roman'); % X轴标柱
subplot(122);surf(x,y,z,'EdgeColor','flat');             % 绘制曲面图，并设定网格线
的颜色属性
xlabel('(b)','Fontsize',14,'Fontname','Times New Roman'); % X轴标柱
```

运行上述程序，输出图形如图 17.8 所示。

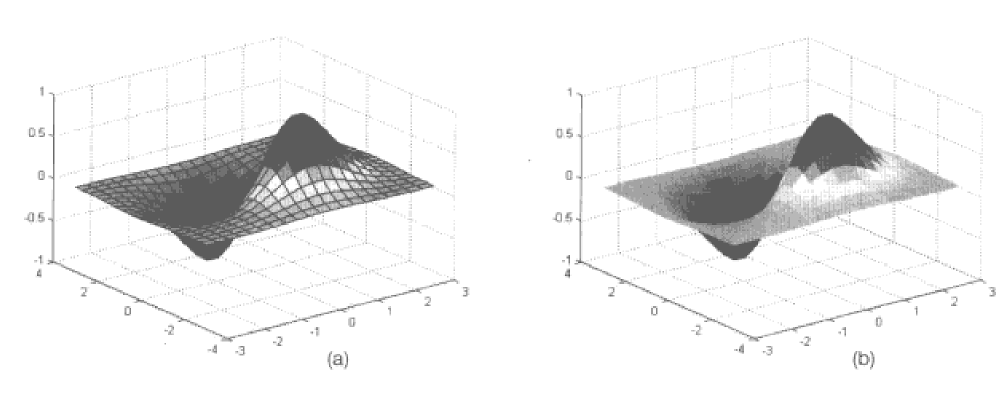


图 17.8　函数 surf 绘制的三维曲面



在图 17.8(b)中黑色网格线的 EdgeColor 属性被设置为“flat”，当函数值的取样点较多时，图(b)中的颜色变换就变得连续起来。

17.1.9 基于数学表达式的三维曲面

函数 ezsurf 可以用来根据数学表达式绘制三维曲面，其调用格式为：

```
ezsurf(funz);
ezsurf(funz,domain);
ezsurf(funx,funy,funz);
ezsurf(funx,funy,funz,domain);
ezsurf(...,n);
ezsurf(...,'circ');
```

参数说明： funx，funy 和 funz 分别是相应坐标轴的函数。domain 是一个含有 2 个或者 4 个元素的向量，用于限制函数表达式中变量的取值范围，其默认值为[-pi*2,pi*2,-pi*2,pi*2]。n 表示变量的取样点数。circ 表示在 domain 参数限定区域内的一个圆形区域中绘制曲面图。

下面举例说明函数 ezsurf 的用法，相关程序如下：

```
funx=inline('sin(s)*t');          % 定义数学表达式
funy=inline('cos(s)*t');          % 定义数学表达式
funz=inline('sinc(t)');           % 定义数学表达式
figure;                           %生成新的图形窗口
subplot(231);ezsurf(@(x,y)funz(x)*cos(y));                    % 绘制曲面图
xlabel('(a)','Fontsize',14,'Fontname','Times New Roman');    % X 轴标柱
subplot(232);ezsurf(@(x,y)sin(x*2)*sinc(y),[-pi,pi]);        % 绘制曲面图，并设定网格线的颜色属性
xlabel('(b)','Fontsize',14,'Fontname','Times New Roman');    % X 轴标柱
subplot(233);ezsurf(funx,funy,funz);                          % 绘制曲面图,并设定网格线的颜色属性
xlabel('(c)','Fontsize',14,'Fontname','Times New Roman');    % X 轴标柱
subplot(234);ezsurf(@(s,t)funx(s,t)*sin(t),@(s,t)funy(s,t)*cos(t^2),@(s,t)funz(t)*sinc(s),[-2,2]);
xlabel('(d)','Fontsize',14,'Fontname','Times New Roman');    % X 轴标柱
subplot(235);ezsurf(@(s,t)funz(t)*sin(s),30);                % 绘制曲面图，并设定网格线的颜色属性
xlabel('(e)','Fontsize',14,'Fontname','Times New Roman');    % X 轴标柱
subplot(236);ezsurf(@(s,t)exp(-[s^2+t^2]),[-2,2],2,'circ');  % 绘制曲面图，并设定网格线的颜色属性
xlabel('(f)','Fontsize',14,'Fontname','Times New Roman');    % X 轴标柱
```

上面的代码给出了利用不同输入函数时的曲面绘制结果（其中利用输入选项 circ 得到在圆形区域内绘图的目的）。运行上述程序，输出图形如图 17.9 所示。

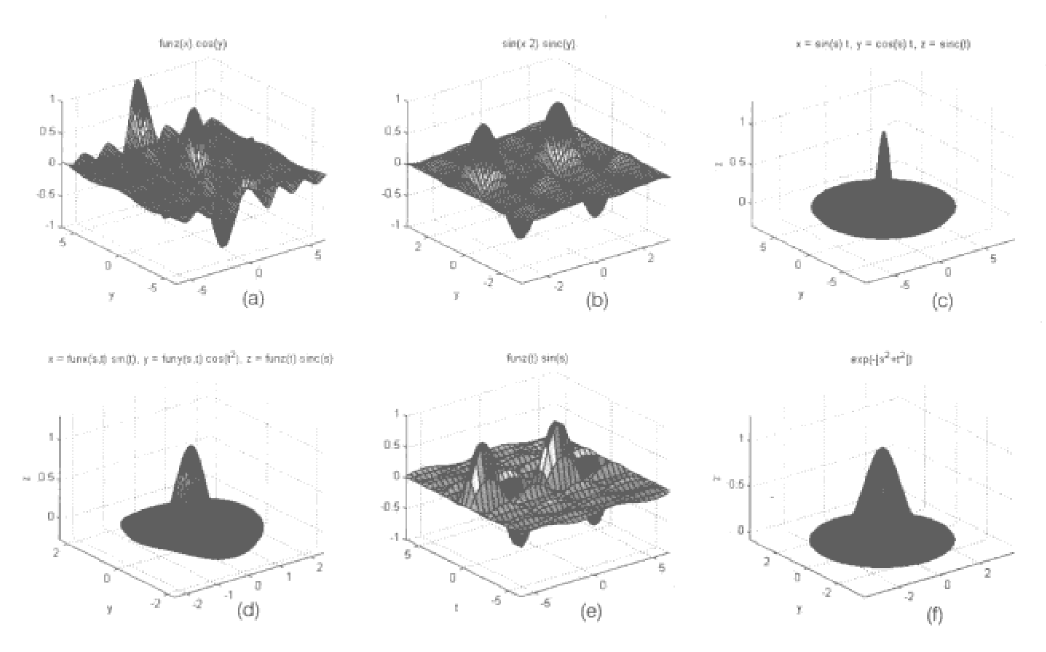


图 17.9 函数 ezsurf 绘制的曲面



函数 ezsurf 在绘制曲面时进行了坐标轴标注和图题注释。

对于函数 ezsurf 曲面属性不能在输入参数中编辑，需要通过函数 ezsurf 的句柄来修改属性值。下面举例说明修改方法。

```
z = inline('y.*(x.^a)./(x.^b + y.^c+0.5)','x','y','a','b','c');   % 定义函数
figure;                                                   %生成新的图形窗口
subplot(121);ezsurf(@(x,y)z(x,y,2,2,4));                  % 直接绘制三维曲面
subplot(122);m=ezsurf(@(x,y)z(x,y,2,2,4));                % 直接绘制三维曲面，同时输出句柄
set(m,'EdgeColor','None');                                % 通过句柄设置曲面上网格线的颜色
```

输出图形如图 17.10 所示，其中图(b)的网格线颜色设置为“None”，即不显示网格线。

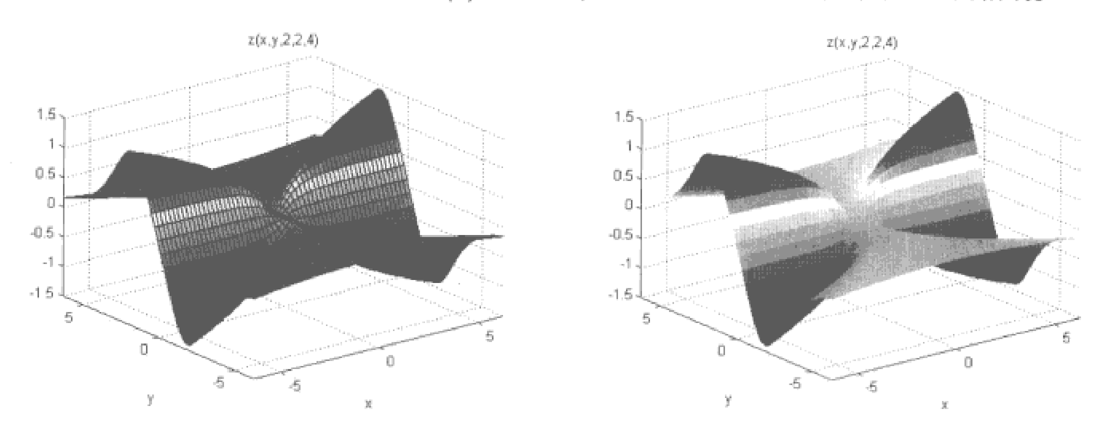


图 17.10　利用句柄修改曲面上网格的颜色

17.1.10　带有等高线的曲面

函数 surfc 可以绘制带有等高线的曲面，其调用格式和函数 surf 相似。这里举例说明其用法：

```
[x,y]=meshgrid(linspace(-4,4,30));            % 生成自变量的采样数值
z =  3*(x-1).^2.*exp(-x.^2 - (y-1).^2) - 8*(x/5 - x.^3 - y.^5).*exp(-x.^2-y.^2) -
1/4*exp(-(x+1).^2 - y.^2);  % 计算函数值
subplot(121);surfc(x,y,z);                    % 三维曲面
xlabel('(a)','Fontsize',14,'Fontname','Times New Roman');      % X 轴标柱
subplot(122);s=surfc(x,y,z,'EdgeColor','k','FaceColor','None');  % 设置曲面的颜色
属性
xlabel('(b)','Fontsize',14,'Fontname','Times New Roman');      % X 轴标柱
```

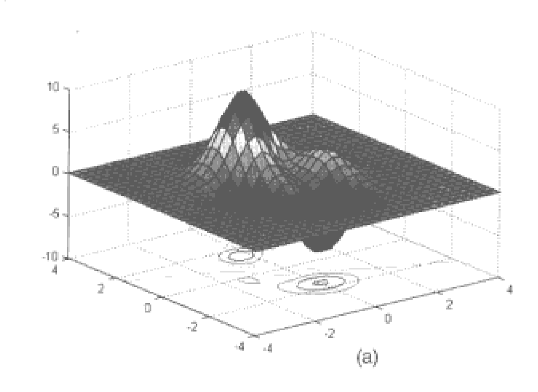
运行上述程序，输出图形如图 17.11 所示。



在图 17.11 中，图(b)的网格线被设置为黑色，而网格内部的填充颜色被设置为“None”，这样曲面就变为“网格”了，即与函数 mesh 绘制的网状图一样。可见函数 ezsurf、函数 surf 与函数 mesh 可以通过属性的设置转化。

函数 ezsurfc 可以根据数学表达式绘制带有等高线的曲面，其调用格式与函数 ezsurf 相似。这

里直接举例说明函数 ezsurfc 的用法。


(a)

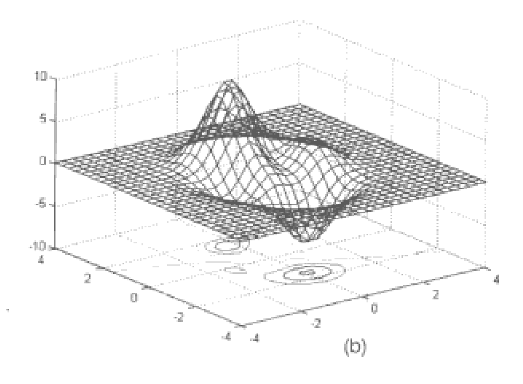

(b)

图 17.11　带有等高线的曲面图

```
figure;                                              %生成新的图形窗口
subplot(121);ezsurfc(@(x,y)sin(x)*exp(-x^2-y^2));  % 绘制带有等高线的三维曲面
subplot(122);ezsurfc(@(x,y)sin(2*y)*exp(-x^2-y^2/2),[-2,2],'circ');  % 绘制曲面，并设置变量范围和圆形区域
view([-111,42])                                      % 设置视角
```

运行上述程序，输出图形如图 17.12 所示。

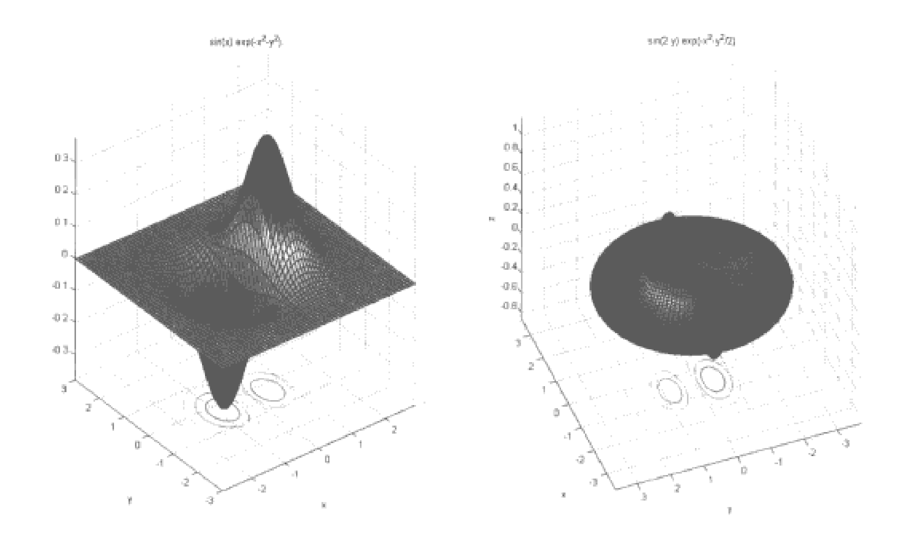


图 17.12　函数 ezsurfc 绘制带有等高线的曲面图形



函数 ezsurfc 属性的修改不能直接在输入参数中进行，需要通过句柄来修改。用户根据前面介绍函数 ezsurf 时给出的例子即可模仿出属性修改的方法。

17.1.11　带有光照效果的曲面

函数 surfl 可以绘制带有光照效果的曲面图，其调用格式为：

```
surfl(Z)
surfl(X,Y,Z)
```

```
surfl(...,'light')
surfl(...,s)
surfl(X,Y,Z,s,k)
```

参数说明： X，Y 和 Z 是曲面关键点的坐标值。light 表示引入光照效果。s 表示光源照射的方向，其为一个含 2 个或者 3 个元素的向量。k 是一个含有 4 个元素的向量，即 k = [ka, kd, ks, shine]，其中 ka 是环境光的贡献，kd 是漫反射系数，ks 是镜面反射系数，shine 镜面发光系数，k 的默认值是[0.55,0.6,0.4,10]。

下面举例说明函数 surfl 的用法，程序如下：

```
figure;                    %生成新的图形窗口
subplot(121);
 [x,y,z] = peaks(30);     % 生成曲面数据
surfl(x,y,z);             % 绘制曲面图
shading interp;           % 着色淡化处理
colormap(gray);axis([-3,3,-3,3]);view(3);    % 设置颜色为灰度格式、坐标轴范围和视角
subplot(122);
surfl(x,y,z,[0.8,0.2,0.8],'light');    % 带有光照效果
shading interp;                  % 着色淡化处理
axis([-3,3,-3,3]);view(3);           % 设置坐标轴范围和视角
```

上面的代码给出了利用函数 surfl 绘制带有光照效果的三维曲面。其中曲面数据是由函数 peaks 得到的，而图 17.13 的左图是利用函数 shading 得到的光滑曲面。运行上述程序，所得图形如图 17.13 的右图所示。

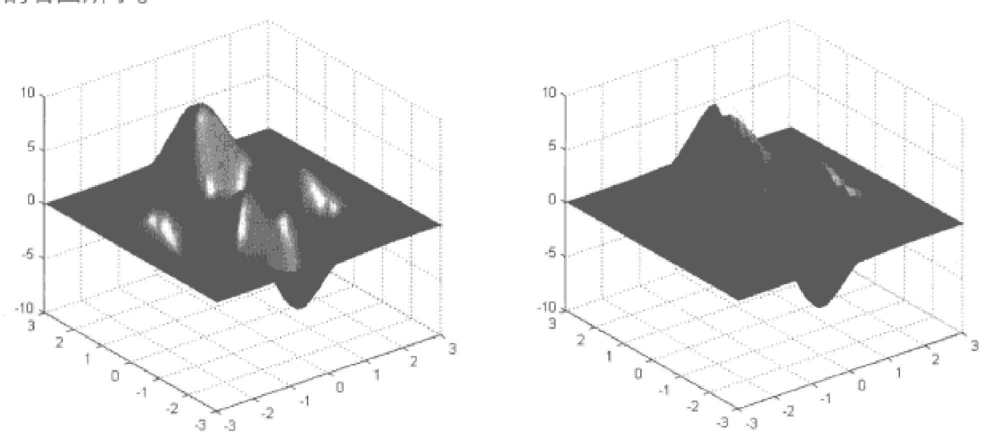


图 17.13 带有光照效果的三维曲面

17.1.12 三维表面法向

函数 surfnorm 可以用来计算或者显示三维表面法向，其调用格式为：

```
[Nx,Ny,Nz] = surfnorm(Z);
[Nx,Ny,Nz] = surfnorm(X,Y,Z);
```

参数说明： Nx，Ny 和 Nz 是返回的方向数据。X，Y 和 Z 表示三维表面的坐标数据。当输出参数默认值，MATLAB 将绘制出含法向线段的曲面。

下面举例说明函数 surfnorm 的用法。

```
[X,Y,Z]=peaks(30);                          % 生成曲面数据
figure;                                     %生成新的图形窗口
surfnorm(X,Y,Z);xlim([-3,3]);ylim([-3,3]);  % 绘制含法线的曲面
```

运行上述程序，输出图形如图 17.14 所示。

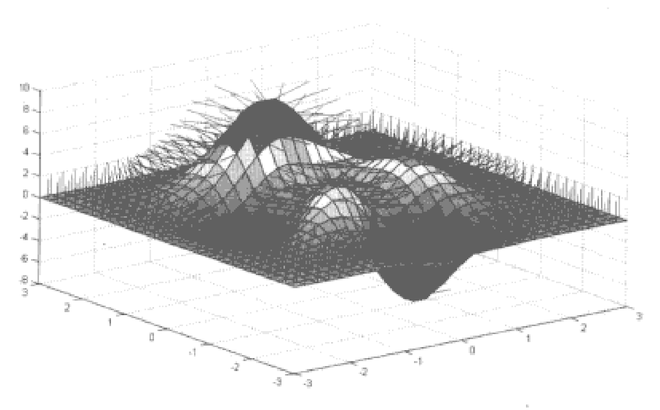


图 17.14　含法线的曲面图

17.1.13　三维等高线

函数 contour3 可以用来绘制三维等高线，其调用格式为：

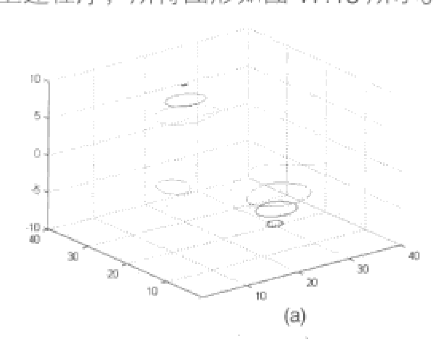
```
contour3(Z)
contour3(X, Y, Z)
```

参数说明：X，Y 和 Z 是相应曲面数据。

例 17–3：调用函数 contour3 绘制三维等高线。

```
figure;                                                          %生成新的图形窗口
subplot(121);contour3(peaks(40));                                % 绘制三维等高线
xlabel('(a)','Fontsize',14,'Fontname','Times New Roman');        % X 轴标柱
subplot(122);mesh(peaks(40),'EdgeColor',[0.85,0.85,0.85]);       % 绘制三维网状曲面，并
把网格线设置为浅灰色
hold on;contour3(peaks(40));                                     % 绘制三维等高线
xlabel('(b)','Fontsize',14,'Fontname','Times New Roman');        % X 轴标柱
```

运行上述程序，所得图形如图 17.15 所示。



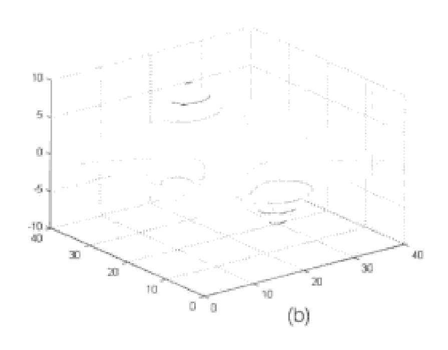


图 17.15　三维等高线

17.1.14 流水效果的曲面

函数 waterfall 可以产生在 X 轴或者 Y 轴方向的具有流水效果的曲面，其调用格式为：

```
waterfall(Z)
waterfall(X,Y,Z)
waterfall(...,C)
```

参数说明： X，Y 和 Z 是曲面采样点的数据。C 用于指定曲面网格线的颜色。

下面的程序调用函数 waterfall 绘制带有流水效果的曲面。

```
[x,y,z] = peaks(40);                                         % 生成曲面数据
figure;                                                      %生成新的图形窗口
subplot(131);waterfall(peaks(40));                           % 绘制具有流水效果的曲面
xlabel('(a)','Fontsize',14,'Fontname','Times New Roman');    % X 轴标柱
subplot(132);waterfall(peaks(40),rand(40));                  % 绘制具有流水效果的曲面
xlabel('(b)','Fontsize',14,'Fontname','Times New Roman');    % X 轴标柱
subplot(133);w=waterfall(x',y',z');                          % 绘制具有流水效果的曲面
set(w,'EdgeColor','k');                                      % 设置网格线为黑色
xlabel('(c)','Fontsize',14,'Fontname','Times New Roman');    % X 轴标柱
```

运行上述程序，所得图形如图 17.16 所示。

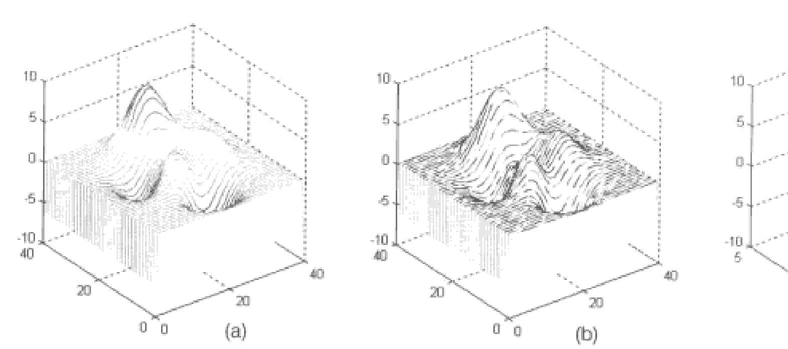


图 17.16 具有流水效果的曲面



在图 17.17 中，图(b)是利用随机矩阵进行着色的（此时网格线为随机的彩色），图(c)利用函数 waterfall 的句柄属性 EdgeColor 设置黑色的网格线。当同时绘制水平和竖直方向的带有流水效果的曲面时就等价于前面介绍的函数 meshz。

17.1.15 颜色表示高度值的图形

函数 pcolor 可以绘制以颜色表示高度值的图形，其调用格式为：

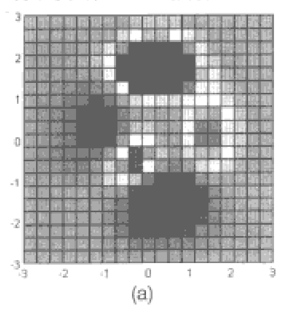
```
pcolor(C)
pcolor(X,Y,C)
```

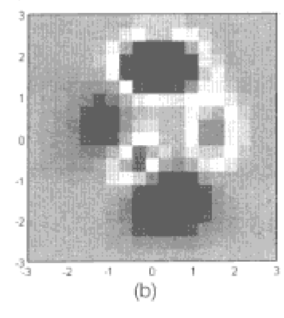
参数说明： C 是一个矩阵。X 和 Y 表示位置的数组，它们可以是向量或者矩阵。函数 pcolor 绘制的图形需要通过句柄来编辑。

下面调用 pcolor 来伪色绘图，程序如下：

```
[x,y,z] = peaks(20);                              % 生成曲面数据
subplot(131);pcolor(x,y,z);                          % 伪色绘图
xlabel('(a)','Fontsize',14,'Fontname','Times New Roman');   % X 轴标柱
subplot(132);p=pcolor(x,y,z);                        % 伪色绘图
set(p,'EdgeColor','flat');                            % 隐去网格线
xlabel('(b)','Fontsize',14,'Fontname','Times New Roman');   % X 轴标柱
subplot(133);hi=image(z);                          % 利用函数绘图作为比对
set(hi,'CDataMapping','scaled');                     % 设置颜色映像属性
xlabel('(c)','Fontsize',14,'Fontname','Times New Roman');   % X 轴标柱
set(gca,'YDir','normal');                            % 设置 Y 轴方向
```

上述程序执行后会得到图 17.17 所示的图形。如果将最后一行语句"set(gca,'YDir','normal');"注释掉，将会得到如图 17.18 所示的一个倒置图形，其中前两个图坐标刻度值与 X 轴标注倒置，第三个图坐标刻度值、X 轴标注和坐标轴内的图案倒置。


(a)


(b)

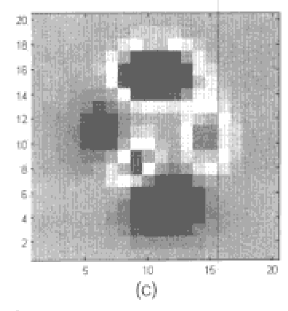
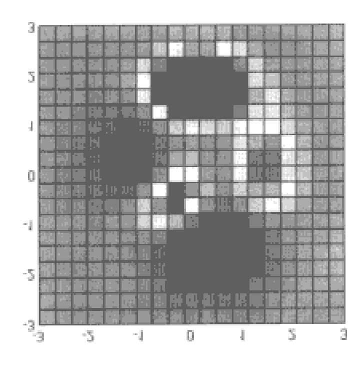
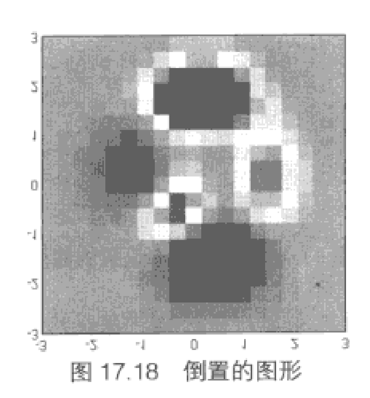

(c)

图 17.17　伪色绘图的结果





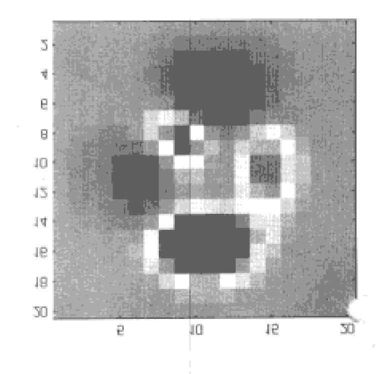


图 17.18　倒置的图形



如果将函数 pcolor 绘制的图形中的网格线隐去（如图 17.17(b)所示），其结果与函数 image 绘制的图形（如图 17.17(c)所示）一致。

前面介绍了三维图形的绘制方法，这里再补充三维图形的切割方法。首先介绍沿水平方向切割曲面的方法。基本思路是对于矩阵 Z，将其中 Z(m,n)>Z0 的元素设置为 Z0 就可以实现切割的效果，下面举例说明切割的效果。

```
[x,y,z]=peaks(800);                                  % 生成离散数据
subplot(131);surf(x,y,z);shading interp;                  % 绘制原函数图像
xlabel('(a)','Fontsize',14,'Fontname','Times New Roman');   % X 轴标柱
z(z>4)=2;     % 挖洞
z(z<-4)=-4;    % 切割
```

```
subplot(132);surf(x,y,z);view(-48,52);shading interp;          % 绘制切、挖操作的结果并
设置视角
xlabel('(b)','Fontsize',14,'Fontname','Times New Roman');  % X 轴标柱
subplot(133);surf(x,y,-z);view(-48,52);shading interp;         % 绘制切、挖操作的结果并
设置视角
    xlabel('(c)','Fontsize',14,'Fontname','Times New Roman');  % X 轴标柱
```

运行上述程序，所得图形如图 17.19 所示。

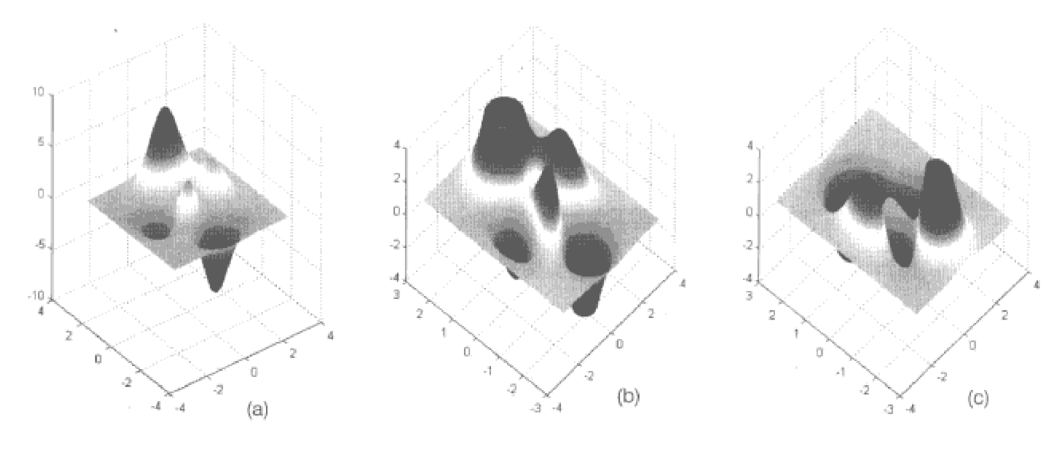


图 17.19 曲面切、挖效果图

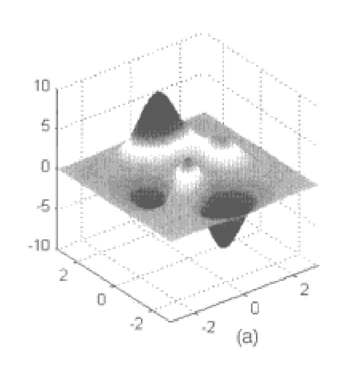


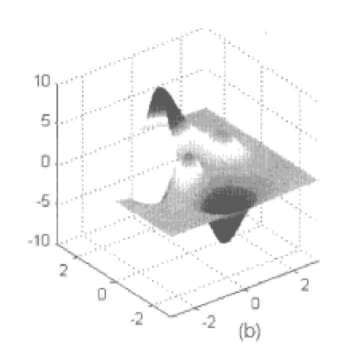
语句“z(z>4)=2;”把大于 4 的曲面部分函数值设置为 2，这样在图形上的效果就会出现一个“平底的坑”。而语句“z(z<-4)=-4;”把曲面上小于-4 的部分切去而留下一个“水平的切面”。图 17.19(c)是上下翻转了图 17.19(b)的结果。

在 XY 平面的切割可以通过把相应位置的元素值设置 NaN 的方式进行。对于 NaN，MATLAB 将不绘图，从而实现了切割的效果，下面举例说明。

```
[x,y,z]=peaks(200);                                          % 生成离散数据
z1=z;z1(x<0&y>0)=nan;                                        % 利用 NaN 切割曲面
z2=z;z2(sqrt(x.^2+y.^2)<1.2)=nan;                            % 利用 NaN 切割曲面
subplot(131);surf(x,y,z);shading interp;xlim([-3,3]);ylim([-3,3]);     % 绘制原函
数图像
xlabel('(a)','Fontsize',14,'Fontname','Times New Roman');        % X 轴标柱
subplot(132);surf(x,y,z1);shading interp;xlim([-3,3]);ylim([-3,3]);    % 绘制切割
操作的结果
xlabel('(b)','Fontsize',14,'Fontname','Times New Roman');        % X 轴标柱
subplot(133);surf(x,y,z2);shading interp;xlim([-3,3]);ylim([-3,3]);    % 绘制切
割操作的结果
xlabel('(c)','Fontsize',14,'Fontname','Times New Roman');        % X 轴标柱
```

运行上述程序，所得图形如图 17.20 所示。





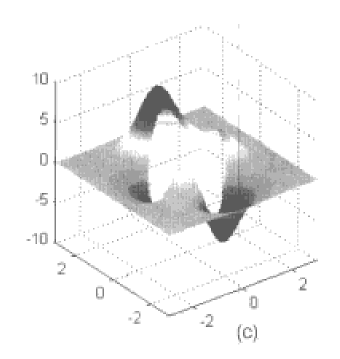


图 17.20　曲面切割结果图



程序中画线部分是切割的关键部分。语句“z1=z;z1(x<0&y>0)=nan;”把曲面 1/4 角内的数据设置为 NaN，从而实现了第二象限内数据的切割；而语句“z2=z;z2(sqrt(x.^2+y.^2)<1.2)=nan;”把距原点距离为 1.2 的圆形区域内点的函数值设置为 NaN，从而实现圆形区域的切割。用户可以根据需要自行定义区域利用特殊值 NaN 来切除。

17.1.16　三维饼图

函数 pie3 可以用来绘制三维饼图，其调用格式为：

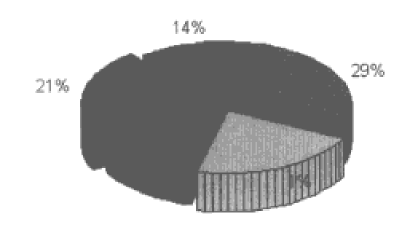
```
pie3(x)
pie3(x,explode)
pie3(...,labels)
```

参数说明： x 表示待统计的数据。explode 用于指定切片是否分离出来，其元素数目与 x 一致，explode 中非零元素表示对应的切片分离。labels 用于重新标柱切片名称。

下面是用函数 pie3 绘制三维饼图的例子。

```
figure;                                                    %生成新的图形窗口
subplot(121);pie3([3,2,3,4,2],[1,0,1,0,0]);                % 绘制饼图
subplot(122);p=pie3([3,2,3,4,2],[1,0,1,0,0],{'A','B','C','D','E'});  % 绘制饼图
set(p(8),'Position',[-1.21866 -0.478151 -0.1]);    % 重新设置标柱字符'B'的位置
set(p(12),'Position',[-0.300403 -1.31615 -0.1]);   % 重新设置标柱字符'C'的位置
```

运行上述程序，所得图形如图 17.21 所示。



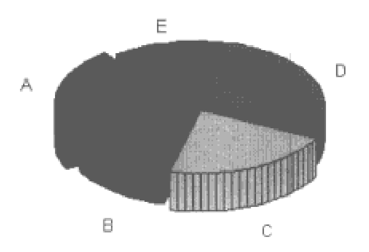


图 17.21　三维饼图



在图 17.21 中，左侧饼图的默认标注是百分比，其中下面两个切块的百分比标注在切块侧面，看不清楚。右侧饼图通过句柄中的 Position 属性把隐藏在切块内的标注移出来。

17.1.17 螺旋体坐标

函数 cylinder 可以生成关于 Z 轴旋转对称的螺旋体坐标，其调用格式为：

```
[x,y,z] = cylinder;
[x,y,z] = cylinder(r);
[x,y,z] = cylinder(r,n)
```

参数说明： x，y 和 z 是螺旋体曲面的坐标。r 用于指定旋转母线的半径，其默认值为[1, 1]。n 用来指定输出坐标的采样点数，其默认值是 20。当函数 cylinder 默认输出参数时，MATLAB 将直接绘制螺旋体。

下面调用 cylinder 函数绘制螺旋体，程序如下：

```
r=2+cos(linspace(0,pi*2));      % 生成半径向量
[x1,y1,z1]=cylinder(r);         % 生成螺旋体坐标
[x2,y2,z2]=cylinder(r,30);      % 生成螺旋体坐标
figure;                         %生成新的图形窗口
subplot(131);cylinder(r);       % 生成三维螺旋体
xlabel('(a)','Fontsize',14,'Fontname','Times New Roman');  % X 轴标柱
subplot(132);surf(x1,y1,z1);    % 生成三维螺旋体
xlabel('(b)','Fontsize',14,'Fontname','Times New Roman');  % X 轴标柱
subplot(133);mesh(x2,y2,z2);    % 生成三维螺旋体
xlabel('(b)','Fontsize',14,'Fontname','Times New Roman');  % X 轴标柱
```

这里给出了利用函数 cylinder 得到螺旋体，利用函数 surfl 和 mesh 绘制旋转曲面的例子。对比可以发现函数 cylinder 和函数 mesh 画出的图形相同。运行上述程序，输出图形如图 17.22 所示。



当输出参数默认时，语句“cylinder(r);”直接得到螺旋体（如图 17.22 (a)所示），其等价于两条语句“[x1,y1,z1]=cylinder(r);”和“surf(x1,y1,z1);”组合的效果（如图 17.22 (b)所示）。图 17.22(c)是指定采样点为 30 的结果。

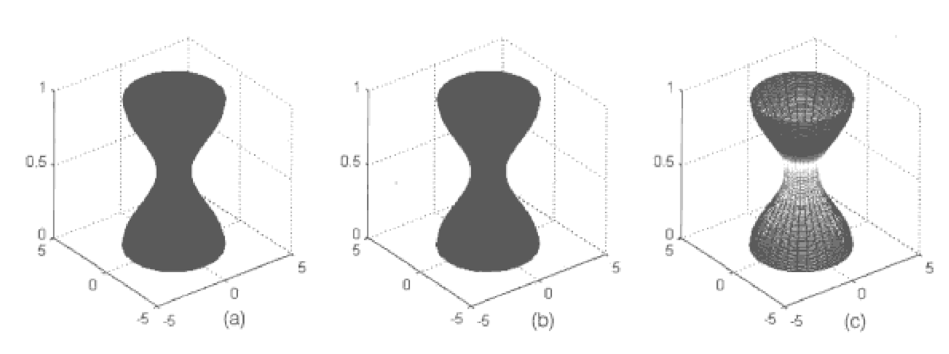


图 17.22 利用函数 cylinder 制作的螺旋体

函数 cylinder 绘制螺旋体的母线是关于 z 的函数，而作者编写了函数 rotationy（该程序保存

在光盘的\Ch17 文件夹中，文件名是 rotationy）来实现母线是 rho（rho=sqrt(x^2+y^2)）的情况对应的旋转体，其调用格式为：

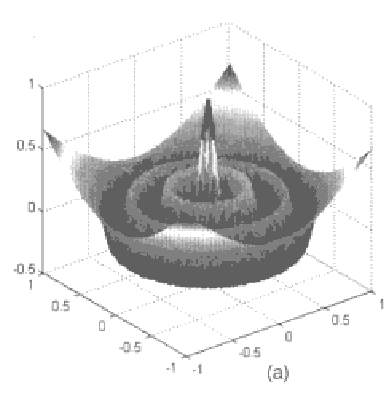
```
[X,Y,Z]=rotationy;
[X,Y,Z]=rotationy(R);
[X,Y,Z]=rotationy(R,N);
```

参数说明： X，Y 和 Z 是生成旋转曲面的坐标。R 是母线，其为极径的函数，默认值为 R=besselj(0,[0:.1:20])。N 表示采样点数，默认值等于 20。函数 rotationy 中用到的插值方法是线性插值法。当输出参数默认时，MATLAB 将直接绘制旋转体的图形。

下面调用函数 rotationy 绘图，计算程序如下：

```
R=sin(linspace(0,pi*2,401));      % 生成母线
[X,Y,Z]=rotationy(R,200,'circ');  % 生成曲面坐标
figure;                           %生成新的图形窗口
subplot(121);rotationy;                                   % 绘图
xlabel('(a)','Fontsize',14,'Fontname','Times New Roman');  % X 轴标柱
subplot(122);surf(X,Y,Z);shading flat;                    % 绘图
xlabel('(b)','Fontsize',14,'Fontname','Times New Roman');  % X 轴标柱
```

运行上述程序，生成图形如图 17.23 所示。



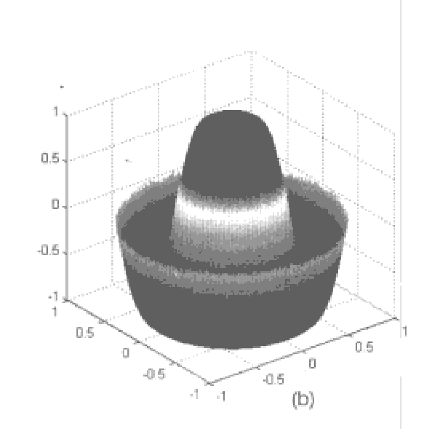


图 17.23　旋转体图形

17.1.18　单位球体的坐标

函数 sphere 用于生成单位球体的坐标，其调用格式为：

```
[x,y,z] = sphere;
[x,y,z] = sphere(n);
```

参数说明： x，y 和 z 表示球面坐标。n 用于指定采样点数，其默认值为 20。当函数 sphere 的输出参数默认时，MATLAB 将直接绘制球面。

下面举例说明该函数的用法。

```
[x,y,z]=sphere;              % 生成球面坐标
figure;                      %生成新的图形窗口
```

```
subplot(121);sphere;     % 直接绘制球面
xlabel('(a)','Fontsize',14,'Fontname','Times New Roman');  % X轴标柱
subplot(122);surf(x,y,z);  % 绘图
xlabel('(b)','Fontsize',14,'Fontname','Times New Roman');  % X轴标柱
```

运行上述程序，输出图形如图 17.24 所示。



图 17.24 分别是利用函数 sphere 直接绘制球面（如图 17.24(a)所示）和利用 surf 函数根据球面坐标绘制的效果（如图 17.24(b)所示），二者结果一样。

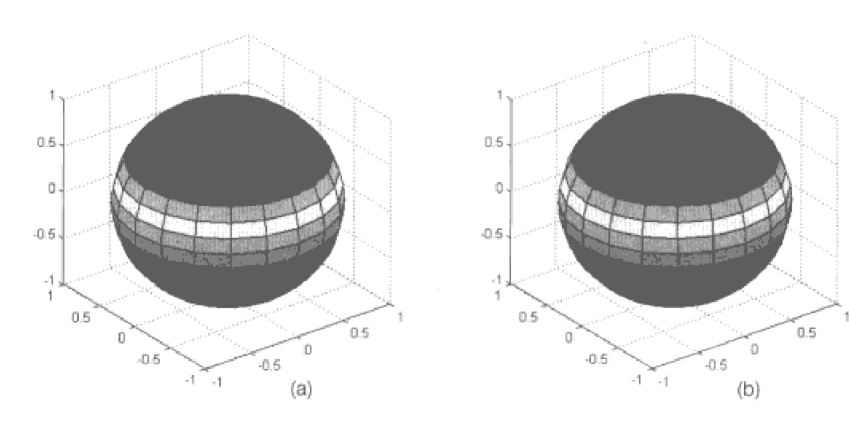


图 17.24 球面的绘制

17.1.19 椭球体表面坐标

函数 ellipsoid 用来生成椭球体表面坐标，其调用格式为：

```
[x,y,z]=ellipsoid(xc,yc,zc,xr,yr,zr);
[x,y,z]=ellipsoid(xc,yc,zc,xr,yr,zr,n);
```

参数说明： x，y 和 z 是椭球面的坐标。xc，yc 和 zc 是椭球中心的坐标。xr，yr 和 zr 是椭球在相应坐标轴上的半径。n 用于定义采样点数，默认值为 20。当输出参数默认时，MATLAB 将直接绘制椭球面。

函数 ellipsoid 的使用与函数 sphere 相似，用户可以参阅前面调用函数 sphere 的例子。

17.1.20 函数 slice

函数 slice 可以用来绘制三维切片图，其调用格式为：

```
slice(x,y,z,v,sx,sy,sz);
slice(x,y,z,v,xi,yi,zi);
slice(v,sx,sy,sz);
slice(v,xi,yi,zi);
slice(...,'method');
```

参数说明： x，y 和 z 表示三维坐标。v 是一个数组，是 x，y 和 z 的函数。sx，sy 和 sz 用于指定切平面的坐标。xi，yi 和 zi 定义完全的切面。method 用于指定内插值的方法，可选值为：linear

（其为默认值）表示使用三次线性内插值法，cubic 表示使用三次立方插值法，nearest 表示使用最近点内插值法。

下面举例说明函数 slice 的用法。

```
[x,y,z] = meshgrid(-2:.2:2, -2:.25:2, -2:.16:2);     % 生成三维空间的坐标
v = x .* exp(-x.^2 - y.^2 - z.^4);                  % 生成三元函数值
[xi,yi]=meshgrid(-2:.1:2);                          % 计算网格坐标
zi=6*exp(-[xi.^2+yi.^2]).*xi;                       % 计算完全切面轴坐标
zi(zi>1)=1;zi(zi<-1)=-1;                            % 限制 zi 取值范围
figure;                                             %生成新的图形窗口
subplot(121);slice(x,y,z,v,[-1.2 .8 2],2,[-2 -.2]);   % 绘制切片图
xlabel('(a)','Fontsize',14,'Fontname','Times New Roman');  % X 轴标柱
subplot(122);slice(x,y,z,v,zi,xi,yi);               % 绘制切片图
xlabel('(b)','Fontsize',14,'Fontname','Times New Roman');  % X 轴标柱
```

上述代码中对右图中的切面数据进行了削顶操作，再调用函数 slice 绘制切片图。运行上述程序的输出结果如图 17.25 所示。其中图(a)使用平面进行切片，图(b)使用曲面进行切片。



在图 17.25 中，图(a)是利用平整的面切三元函数的曲面，而图(b)利用的则是弯曲的面。

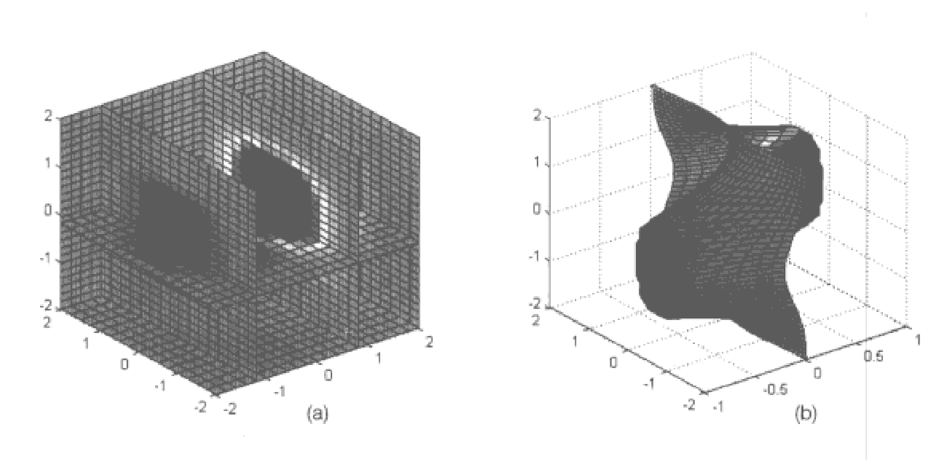


图 17.25　三维切片图

17.2　彩色图及颜色条

前面介绍的函数 surf 和 surfc 等可以得到彩色的图形，对于彩色图形的修改可以得到不同的视觉效果，利用好彩色图像的着色技术可以增加图形的表现力。

17.2.1　控制着色方式

MATLAB 提供了函数 colormap 控制着色方式，该函数调用格式为：

```
colormap(map)
colormap('default')
```

```
colormap('stylename')
```

参数说明： map 是一个 3 列矩阵，其元素数值定义于区间[0,1]。矩阵的每行元素表示 1 个真彩色向量，即红、绿、蓝 3 基色的系数。default 用于设置当前彩色图为默认色图。stylename 表示 MATLAB 提供的预定义的色图样式名称，其具体取值如表 17.2 所示。

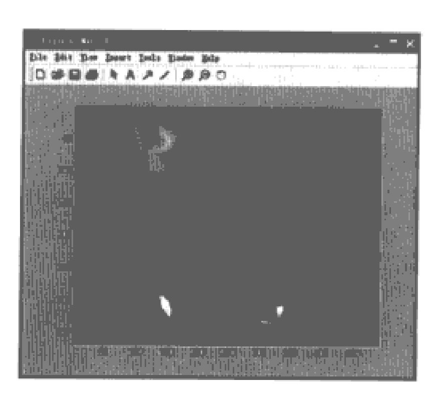
表 17.2　MATLAB 中预定义的色图样式

色图名称	说明	色图名称	说明
autumn	平滑的红、橘黄、黄色	jet	蓝色为头、红色为尾的饱和色
bone	高蓝色灰度渐进	lines	多线绘制时的配置色
colorcube	三纯色浓淡交错	pink	淡粉红色图
cool	青色、品红色浓淡交错	prism	光谱交错色图
copper	纯铜色线性	spring	青黄浓淡色图
flag	红、白、蓝、黑色交错	summer	绿黄浓淡色图
gray	灰度渐进	winter	蓝绿浓淡色图
hot	黑、红、黄、白色浓淡交错	white	纯白色图
hsv	两端为红色的饱和色		

下面给出色图着色方案控制的例子：

```
load spine;                              % 导入图像数据
figure; image(X); colormap bone;    % 显示图像，并设置着色方案为 bone
figure; image(X); colormap('hot');    % 显示图像，并设置着色方案为 hot
```

运行上述程序，输出图像如图 17.26 所示。



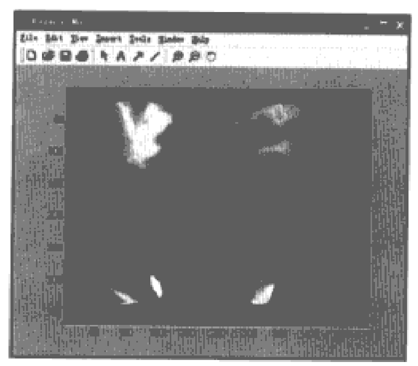


图 17.26　着色方案测试结果



函数 colormap 将对图形窗口内的所有图形着色，因此需要在不同图形窗口中才能得到不同的着色方案。

17.2.2　图片亮度的控制

函数 brighten 可以实现对图片明暗的控制，其调用格式为：

```
brighten(beta);
```

参数说明： beta 是一个定义于[-1, 1]区间内的数值，其中 beta 在[0, 1]范围内时色图较亮。

下面的语句可以测试明暗的差别：

```
figure;image(X);colormap bone;brighten(0.6)
figure;image(X);colormap bone;brighten(-0.6)
```

上述程序得到如图 17.27 所示的结果，读者可以比较二者的亮度与输入参数的关系。

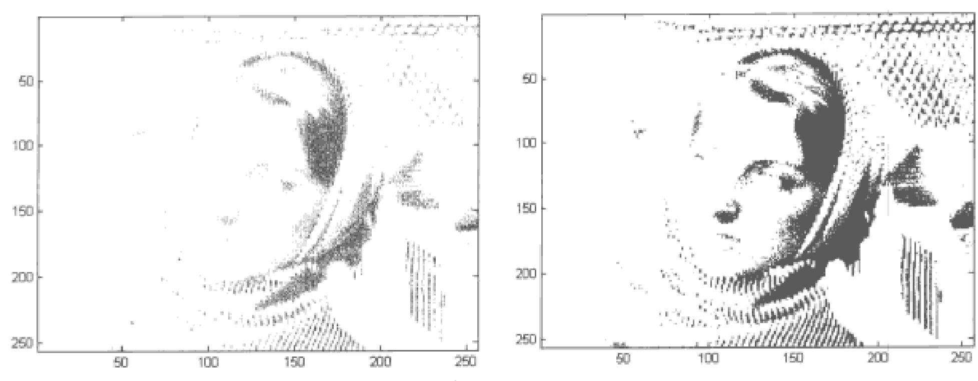


图 17.27　不同亮度的比较

17.2.3　绘制色轴

函数 colorbar 用于绘制色轴，即各种颜色对应的数值，其调用格式为：

```
colorbar;
colorbar('Location');
```

参数说明： Location 用于指定色轴的位置，其可选值为：North 对应于坐标轴内部上侧，South 对应于坐标轴内部下侧，East 对应于坐标轴内部右侧，West 对应于坐标轴内部左侧，NorthOutside 对应于坐标轴外部上侧，SouthOutside 对应于坐标轴外部下侧，EastOutside 对应于坐标轴外部右侧，WestOutside（其为默认值）对应于坐标轴外部左侧，这些位置关系对应于地图上的方位关系。

17.2.4　指定色轴的刻度

函数 caxis 用于指定色轴的刻度范围和比例，其调用格式为：

```
caxis(v)
caxis('manual')
caxis('auto')
```

参数说明： v 是一个含 2 个元素的向量，用于指定色轴的取值范围。manual 用于指定在当前范围内的色轴比例。Auto 表示设置自动对应于数值范围。

下面举例说明函数 colorbar 和函数 caxis 的用法，程序如下：

```
figure;                %生成新的图形窗口
colorbar('North');     % 把色轴放置在坐标轴内部上侧
colorbar('South');     % 把色轴放置在坐标轴内部下侧
colorbar('East');      % 把色轴放置在坐标轴内部右侧
colorbar('West');      % 把色轴放置在坐标轴内部左侧
caxis([0,2]);          % 设置色轴范围
```

```
colorbar('NorthOutside');  % 把色轴放置在坐标轴外部上侧
colorbar('SouthOutside');  % 把色轴放置在坐标轴外部下侧
colorbar('EastOutside');    % 把色轴放置在坐标轴外部右侧
colorbar('WestOutside');    % 把色轴放置在坐标轴外部左侧
```

上述程序运行后可以得到如图 17.28 所示的图形。

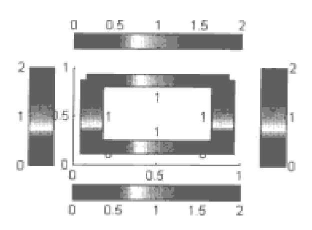


图 17.28 色轴的绘制



从图 17.28 中可以看出函数 caxis 对其前后调用函数 colorbar 绘制的色轴的数值范围都是有限制的。

17.2.5 图形的映像数据表

函数 rgbplot 可以用来绘制图形的映像数据表，其调用格式为：

```
rgbplot(map)
```

参数说明：map 是颜色映像，是 3 列矩阵。函数 rgbplot 以红、绿、蓝 3 种颜色绘制相应的 3 列数据。

下面举例说明函数 rgbplot 的用法。

```
figure;                                                    %生成新的图形窗口
load woman;subplot(131);rgbplot(map);axis square;        % 绘制 woman.mat 数据中的颜
色映像
xlabel('(a)','Fontsize',14,'Fontname','Times New Roman');  % X 轴标柱
load spine;subplot(132);rgbplot(map);axis square;          % 绘制 spine.mat 数据中的
颜色映像
xlabel('(b)','Fontsize',14,'Fontname','Times New Roman');  % X 轴标柱
subplot(133);rgbplot(jet);axis square;                    % 绘制 jet 着色方式的颜色映像
xlabel('(c)','Fontsize',14,'Fontname','Times New Roman');   % X 轴标柱
```

运行上述程序，输出图形如图 17.29 所示。

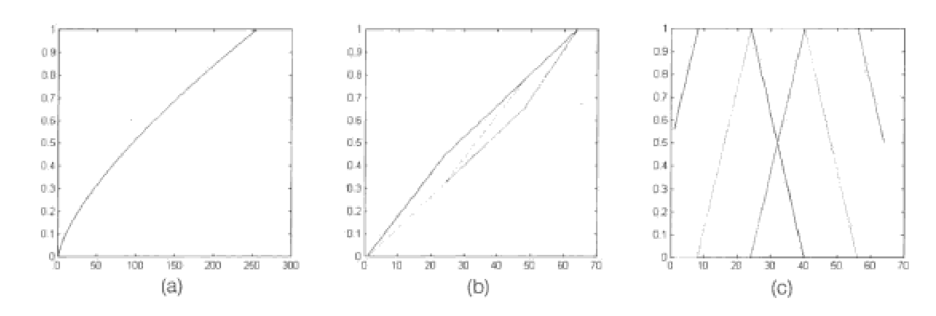


图 17.29 函数 rgbplot 绘制的图形

17.2.6 设置颜色渲染属性

函数 shading 可以用来设置图形的颜色渲染属性，调用格式为：

```
shading
shading flat
shading interp
shading faceted
```

参数说明：flat 表示网格线与网格表面颜色相同，颜色取线段两端或者网格中下标最小点的颜色。Interp 表示网格线和网格表面的颜色由端点和顶点的颜色经插值计算获得。faceted（其为默认值）表示在 flat 着色方式的基础上，由网格四周勾画黑色网格线。

下面举例说明函数 shading 的 3 种着色方式的效果。

```
figure;                                                    %生成新的图形窗口
subplot(131);peaks(40);shading flat;                       % 绘图并以 flat 方式进行颜色渲染
xlabel('flat','Fontsize',14,'Fontname','Times New Roman');     % X 轴标柱
subplot(132);peaks(40);shading interp;                     % 绘图并以 interp 方式进行颜色渲染
xlabel('interp','Fontsize',14,'Fontname','Times New Roman');  % X 轴标柱
subplot(133);peaks(40);shading faceted;                    % 绘图并以 faceted 方式进行颜色渲染
xlabel('faceted','Fontsize',14,'Fontname','Times New Roman'); % X 轴标柱
```

运行上述程序，输出图形如图 17.30 所示。

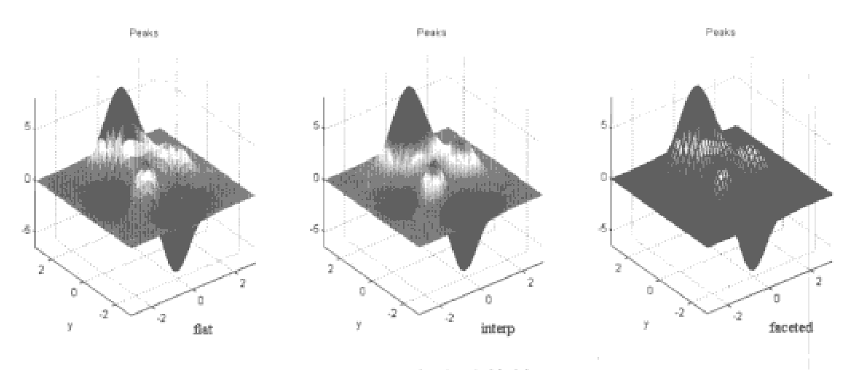


图 17.30　颜色渲染结果



在图 17.30 中，当绘图数据中采样点较多时，图(a)将和图(b)相近。

17.2.7 透明度的设置

可以通过绘图对象的属性 FaceAlpha 和 EdgeAlpha 来设定曲面对象的透明度，此外用户还可以利用函数 alpha 设置当前坐标轴内绘图对象的透明程度，该函数调用格式为：

```
alpha(v)
alpha('x')
alpha('y')
```

```
alpha('z')
alpha('rand')
alpha('scaled')
alpha('direct')
```

参数说明： v是一个介于0和1之间的数，用于控制图形对象的透明程度，当v较小时透明性较好。x，y和z用于设定相应方向为透明。rand用于设定随机透明。scaled和direct用于指定透明性的映像数据类型。

下面举例说明透明性的设定。

```
figure;                                                    %生成新的图形窗口
subplot(131);surf(peaks,'FaceAlpha',0.3,'EdgeAlpha',0.3);   % 绘图并指定面和线的透
明参数
subplot(132);surf(peaks);alpha(0.4);                       % 绘图，设定坐标轴内图形的透明程度
subplot(133);surf(peaks);alpha('rand');                     % 绘图，设定坐标轴内图形的透明程
度为随机
```

运行上述程序，所得图形如图17.31所示。

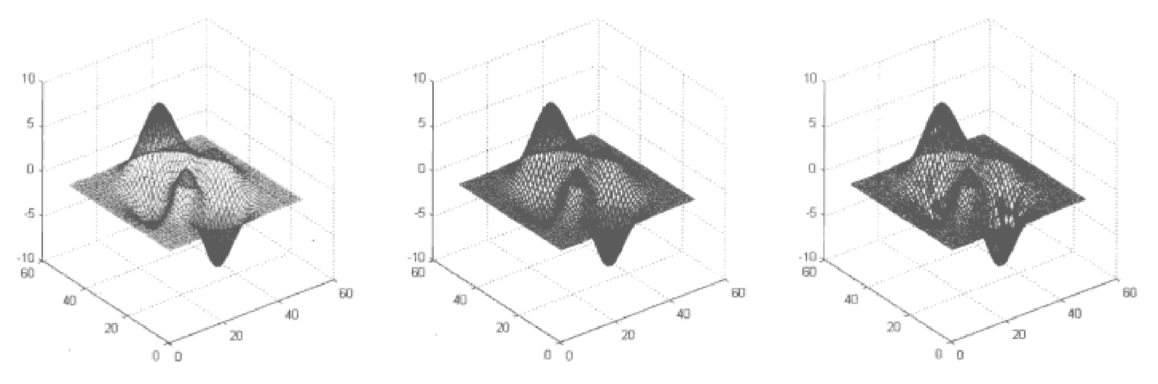


图17.31　设定图形对象为不同的透明程度



这里使用了图形对象属性FaceAlpha和EdgeAlpha以及函数alpha来设定曲面的透明度。

17.2.8　单色网格曲面

有时需要得到黑色网格线对应的曲面，从前面介绍的例子中，知道通过设置图形对象的EdgeColor属性可以使网格线颜色变为黑色。本节再介绍利用函数colormap来设置黑色网格线的方法，即通过下面的语句实现：

```
[X,Y,Z]=peaks(30);      % 生成曲面数据
figure;                 %生成新的图形窗口
mesh(X,Y,Z);            % 绘制网状图
colormap([0,0,0]);      % 设置为黑色映像关系
```

运行上述程序，输出图形如图17.32所示。

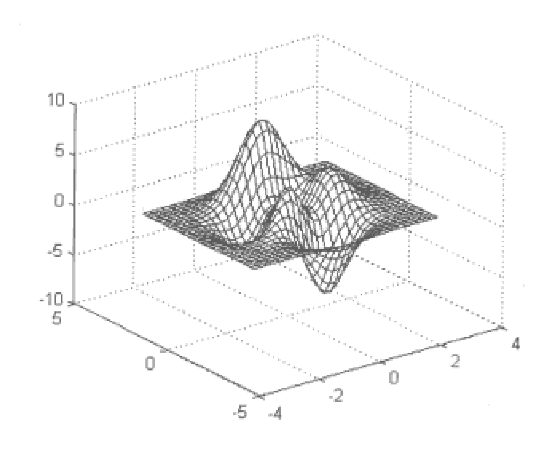


图 17.32　黑色网格线的网状图



用户可以在函数 colormap 中设置其他含 3 个元素的向量以得到其他颜色的网格线。需要注意的是绘制这样的网格线图时，采样点数据适中即可。如果网格线密集则显示不出来曲面的脉络，稀疏则会使精度降低。

前面介绍的函数 waterfall 绘制的曲面带有下面的流水效果部分，这里考虑去掉流水效果，即绘制由单向线条组成的曲面图。作者编写了函数 meshxy（其保存于光盘中\Ch17 文件夹下的 meshxy.m 文件）来实现这个功能，程序实现的思想是依次调用函数 plot3 绘制三维曲线。该函数调用格式为：

```
meshxy(X,Y,Z,c,d);
```

参数说明： X，Y 和 Z 表示曲面的数据，这里要求 X，Y 和 Z 是 3 个相同大小的矩阵。c 表示取线的颜色，可以是特殊颜色，也可以用 3 个元素定义的颜色来表示。d 是表示方向的字符串，可选值为 x 和 y，其中 x 表示 X 轴上的网格线，y 表示 Y 轴上的网格线。

下面举例说明函数 meshxy 的用法。

```
[X,Y,Z]=peaks(30);                    % 生成曲面的离散数据
[Xi,Yi]=meshgrid(-3:.1:3,-2:.1:2);         % 生成离散采样坐标
Zi=Yi.*exp(-[Xi.^2+Yi.^2]);           % 计算二元函数值
subplot(121);meshxy(X,Y,Z,'k','x');       % 绘制 X 轴上的黑色线条曲面
xlim([-3,3]);ylim([-3,3]);              % 设置数据范围
xlabel('(a)','Fontsize',14,'Fontname','Times New Roman');  % X 轴标柱
subplot(122);meshxy(Xi,Yi,Zi,'r','y');     % 绘制 Y 轴上的红色线条曲面
xlim([-3,3]);ylim([-2,2]);              % 设置数据范围
xlabel('(b)','Fontsize',14,'Fontname','Times New Roman');  % X 轴标柱
```

上面的程序利用函数 meshxy 和两个输入参数绘制一个方向的网状图，其中分别给出了 X 轴方向和 Y 轴方向的曲面图。运行上述程序，所得图形如图 17.33 所示。

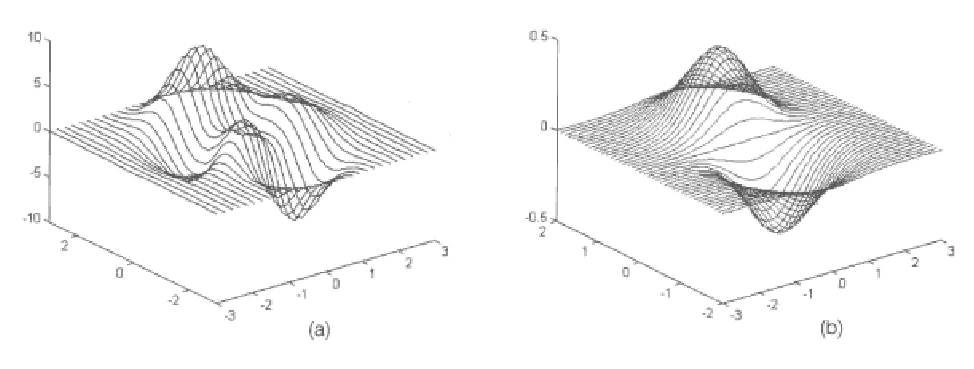


图 17.33 由单向线条组成的网状图

对于函数 meshxy 的图形，同样可以利用 NaN 切割。把下面两条语句加到前面程序中语句“subplot(121);meshxy(X,Y,Z,'k','x');”之前，执行后即可得到如图 17.34 所示的具有切割效果的图形。

```
Z(X<0&Y>0)=NaN;
Zi(Xi<0&Yi<0)=NaN;
```

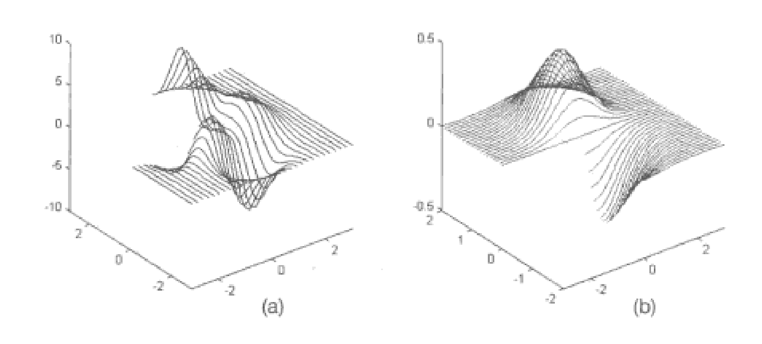


图 17.34 具有切割效果的网状图

17.3 视角与光照

本节来介绍视角和光照方面的函数用法，利用它们可以进一步加强三维图形的表现力。

17.3.1 改变三维图形的视角

用户可通过函数 view 和鼠标两种方式来改变三维图形的视角。函数 view 的调用格式为：

```
view(AZ,EL)
view([AZ,EL])
view([X, Y, Z])
view(2)
view(3)
```

参数说明： 参数 AZ 和 EL 分别为方位角（Azimuth）和仰角（Elevation）。X，Y 和 Z 用于设置笛卡儿坐标系视角，向量[X, Y, Z]对应的单位方向矢量起作用（MATLAB 忽略了该向量的振幅）。2 表示设定图形对象为二维形式，即从 Z 轴上方向下看（即俯视图，其中 AZ=0，EL=90）。3 是函

数 view 的默认值，此时 AZ=-37.5，EL=30。

下面举例说明函数 view 的用法。

```
[x,y,z]=peaks(30);                                              % 生成绘图数据
figure;                                                         %新建图形窗口
subplot(131);mesh(x,y,z);view(-49,36);xlim([-3,3]);ylim([-3,3]);  % 绘图并设定视角和坐标范围
xlabel('(a)','Fontsize',14,'Fontname','Times New Roman');     % X 轴标柱
subplot(132);surf(x,y,z);view([3,2,1]);xlim([-3,3]);ylim([-3,3]);  % 绘图并设定视角和坐标范围
xlabel('(b)','Fontsize',14,'Fontname','Times New Roman');     % X 轴标柱
subplot(133);surf(x,y,z);view(2);xlim([-3,3]);ylim([-3,3]);       % 绘图并设定视角和坐标范围
xlabel('(c)','Fontsize',14,'Fontname','Times New Roman');     % X 轴标柱
```

运行上述程序，输出图形如图 17.35 所示。

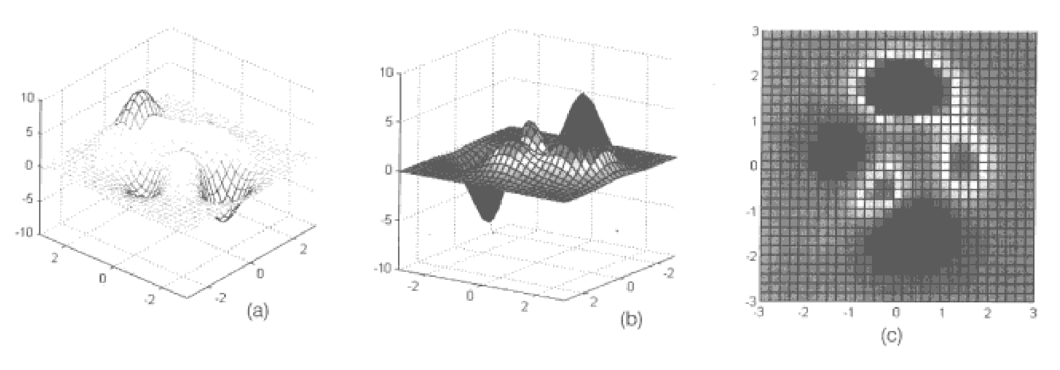


图 17.35　不同视角下的曲面图形

除了前面介绍的函数 view，用户还可以利用鼠标来手动设置坐标轴的视角。具体做法如下：首先单击图形窗口中 3 维旋转按钮（Rotate 3D），如图 17.36 所示；然后在坐标轴窗口内单击鼠标左键就会出现一个虚线圆圈，按住左键不放移动鼠标就可以旋转坐标轴，此时窗口左下角位置会出现 AZ 和 EL 的实时取值；用户判断当前坐标轴视角是否合适，如果满足要求了，释放左键就可以选定当前视角了。利用这种方法确定单个坐标轴的视角可以比较快地找到合适的视角。

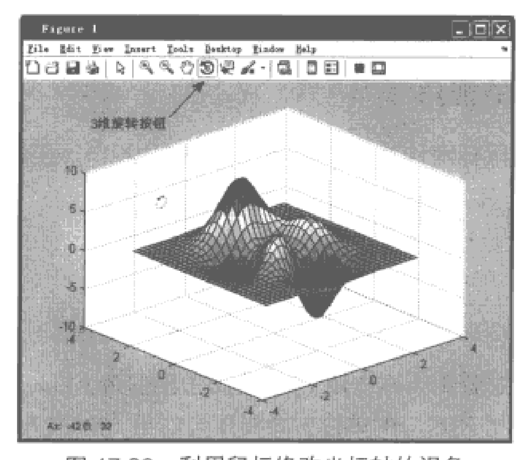


图 17.36　利用鼠标修改坐标轴的视角

17.3.2 灯光效果设置

MATLAB 提供了 camlight，light，ligtangle，lighting，material 等函数设置灯光效果，下面介绍这几个函数的用法。

```
camlight headlight
camlight right
camlight left
camlight(AZ, EL)
camlight(...,'style')
```

参数说明：headlight，right，left 表示创建相对于摄像方位的灯光，它们分别表示上方、左侧和右侧方位。AZ 和 EL 表示灯光的方位角（Azimuth）和仰角（Elevation）。style 表示灯光的类型，可取值为：local（其为默认值）表示灯光在各个方向都有辐射光；infinite 表示光源发出平行光，即光源置于无限远处。

函数 light 用于修改当前灯光对象的属性，其调用格式为：

```
light('PropertyName', PropertyValue, ...)
```

参数说明：参数 PropertyName 和 PropertyValue 分别是灯光对象的属性和属性值。该函数的属性有：Position 表示灯光的位置，其为 3 个元素的向量，默认值为[1,0,1]；Color 表示灯光的颜色，默认为白色；style 表示光源的位置，可选值为 infinite 无穷远（其为默认值）和 local 近处。

函数 lightangle 用于设定光源的方位角，其调用格式为：

```
lightangle(AZ, EL)
```

参数说明：AZ 和 EL 表示灯光的方位角（Azimuth）和仰角（Elevation）。

函数 lighting 用于指定光源照明的模式，其调用格式为：

```
lighting flat
lighting gouraud
lighting phong
lighting none
```

参数说明：flat（其为默认值）表示光线均匀地照射到图形面上。gouraud 表示对顶点颜色插补，然后对顶点勾画的面色进行插补（常用于曲线表现）。phong 表示对顶点法线插值后计算像素的反色。none 表示关闭光源。

函数 material 用来控制光照效果的材质，其调用格式为：

```
material shiny
material dull
material metal
material([ka, kd, ks])
material([ka, kd, ks, n])
material([ka, kd, ks, n, sc])
```

参数说明：shiny 表示光照比较明亮。dull 表示光照暗淡。metal（为默认值）表示光照带有金属光泽。Ka，kd，ks，n，sc 表示反射过程的系数，具体含义为：ka 表示各向同性均匀的背景光的强度，kd 表示漫反射强度，ks 表示硬反射系数，n 表示控制镜面亮点大小的镜面指数，sc 表示

镜面颜色的反射系数。

下面举例说明上面介绍的函数用法。

```
subplot(141);surf(peaks(30));camlight(43,120);                % 绘图并设置光源位置
xlabel('(a)','Fontsize',14,'Fontname','Times New Roman');     % X轴标柱
subplot(142);surf(peaks(30));light('Color','r');              % 绘图并设置光源颜
色
xlabel('(b)','Fontsize',14,'Fontname','Times New Roman');     % X轴标柱
subplot(143);surf(peaks(30));lighting('phong');               % 绘图并设置插值算法
xlabel('(c)','Fontsize',14,'Fontname','Times New Roman');     % X轴标柱
subplot(144);surf(peaks(30));material dull;                   % 绘图并设置光照效果材质
xlabel('(d)','Fontsize',14,'Fontname','Times New Roman');     % X轴标柱
```

上述程序运行后得到图形如图 17.37 所示。

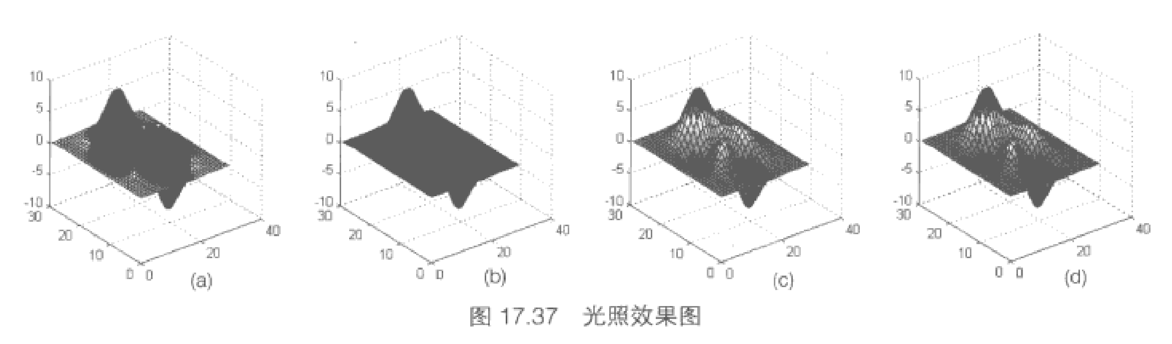


图 17.37　光照效果图

17.4　图形的注释

对于三维图形的注释可以使用的函数有 text，xlabel，ylabel，zlabel 和 legend 等，这些函数的调用格式在第 16 章已经介绍过，这里直接举例说明这几个函数的用法。

下面一段程序给出了不同类型的标注。

```
t=linspace(0,pi,201);                 % 变量离散采样
x1=cos(t*8);y1=sin(t*8);z1=t;         % 生成绘图数据
x2=cos(t*8)/2;y2=sin(t*8)/2;z2=t;     % 生成绘图数据
plot3(x1,y1,z1,'k');hold on;          % 绘制三维曲线
plot3(x2,y2,z2,'r:');                 % 绘制三维曲线
legend('C1','C2',1);                  % 图例标注
xlabel('{\itx} (mm)','Fontname','Times New Roman','Fontsize',14,'Rotation',17);
% 标注 X 轴
ylabel('{\ity} (mm)','Fontname','Times New Roman','Fontsize',14,'Rotation',-30);
% 标注 Y 轴
z=zlabel('{\itz} (mm)','Fontname','Times New Roman','Fontsize',14);          %
标注 Z 轴
title('3D curves','Fontname','Times New Roman','Fontsize',14);               %
标注图题
text(-0.5,0.5,4,'3D {\itcircle}','Fontname','Times New Roman','Fontsize',14);
% 注释文字
```

首先程序给出了三维曲线的绘制，然后调用函数 legend 添加图例、调用函数 xlabel 以及 ylabel 和 zlabel 添加坐标轴标注、调用 title 函数添加图题、调用函数 text 在图中加字符标注。运行上述程

序，输出如图 17.38 所示的图形。

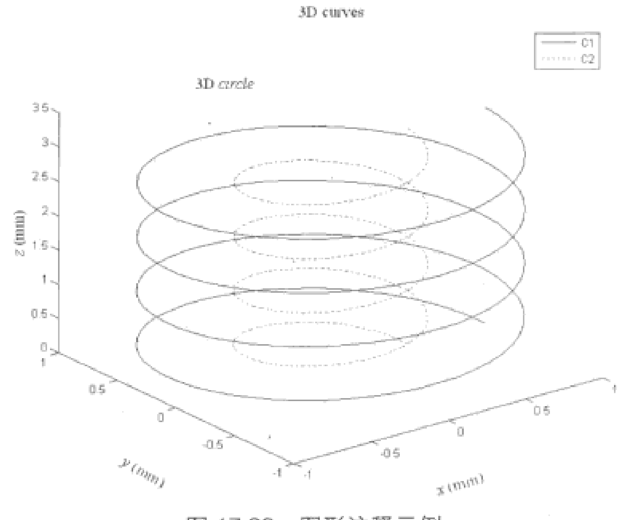


图 17.38 图形注释示例

在用户调用函数 gtext 利用鼠标在三维坐标轴内进行标注时，调用下面语句：

```
gtext('3D {\itcircle}','Fontname','Times New Roman','Fontsize',14);     % 鼠标注释文字
```

MATLAB 将会提示用户：

```
??? Error using ==> gtext at 39
View must be two-dimensional.
```

如果用户把函数文件 gtext.m（可用语句 edit gtext 打开该文件）中的 37~40 行语句打开，看到：

```
[az el] = view;
if az ~= 0 || el ~= 90
    error('MATLAB:gtext:InvalidView', 'View must be two-dimensional.')
end
```

就表示函数 gtext 可以在三维坐标轴中注释文字内容了。通过可编辑按钮（Edit plot）可以调整函数 gtext 标注文字的字体、字号和位置等属性。

斜体希腊字母的注释可以利用控制字符“\it”和希腊字母的控制符（具体内容可参见表 16.9），语法格式为：

```
text(x,y,z,'{\it\beta}')
```

此时得到斜体希腊字母的图形通过拷屏的方式可以得到斜体的希腊字母，但是如果是通过 saveas 和 print 来复制得到图形，其斜体希腊字母将变为直体。另外一个得到斜体希腊字母的途径是利用完整版 Adobe Acrobat 软件提供的打印机“Adobe PDF”来打印当前图形窗口中的内容为 PDF 文件。在利用 Adobe Acrobat 软件把 PDF 文件转为 eps 或者 tiff 等文件格式时，所得图像文件在四周存在空白区域，可以通过 Adobe Acrobat 软件提供的“剪裁页面”菜单功能将其切去。

17.5 小结

本章主要介绍了三维图形的绘制方法。MATLAB 提供了很多关于三维绘图方面的函数，利用它们可以方便地绘制三维图形。首先介绍了三维网状图、曲面图以及特殊形式的三维图形（如饼图、旋转体、流水效果的图形和等高线等）的绘制方法。接下来介绍了彩色图形的颜色管理以及色轴的添加与修改方法，它们对于增加图形表现力具有重要作用。最后还介绍了单色网格曲面的绘制、控制视角和光照的方法，以及三维图形的标注方法。

◆ ◆ ◆

第 18 章　用户图形界面设计

本章包括

- **菜单设计**　介绍不同类型菜单条的设计方法。
- **自定义工具条**　介绍工具条的制作和按钮的设计方法。
- **控件设计**　介绍利用函数 uicontrol 设计不同控件的方法。
- **对话框**　介绍不同类型对话框的创建过程。
- **实例**　介绍利用 GUIDE 工具设计图形界面窗口，同时给出 GUI 实例。

图形用户界面（Graphical User Interfaces，简称 GUI）是 MATLAB 提供的人机交互操作的一个工具和方法，本章主要介绍这方面的知识。GUI 是包含基本图形对象，如坐标轴窗口、按钮、编辑框文本框、滑动条等控件的用户操作界面。通过选择不同的图形对象、设置其关键属性，可以引起不同的动作，比如利用鼠标在坐标轴取点、单击鼠标执行不同的动作等。一个比较好的 GUI 作品可以很直观地让用户进行交互式操作。MATLAB 提供了比较全面的空间类型来组建 GUI。同时在 GUI 环境中还可以调用其他 MATLAB 计算程序，从而结合 MATLAB 强大的计算功能。

18.1　菜单设计

除了 MATLAB 自带的图形窗口菜单，如图 18.1 所示，用户还可以创建新的菜单。MATLAB 提供了 uimenu，menu 和 uicontextmenu 等函数来制作不同形式的菜单。下面详细介绍这些函数的用法。

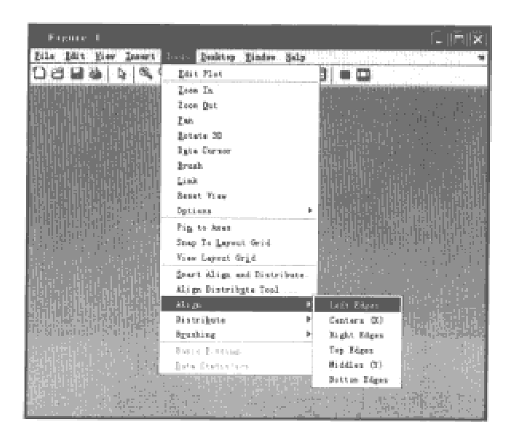


图 18.1　图形窗口的界面

18.1.1　函数及使用说明

函数 uimenu 可以在图形窗口内添加菜单条，其调用格式为：

```
uimenu(H,'PropertyName1',value1,'PropertyName2',value2,...)
```

参数说明：H 是 Figure 窗口或者菜单条对象的句柄。PropertyName1 和 PropertyName2 是菜单条对应的属性名称，value1 和 value2 是属性相应的取值。

函数 uimenu 的常用属性如表 18.1 所示。

表 18.1　uimenu 的属性说明

属性名称	说明	属性名称	说明
Checked	菜单检查标记，默认值为 off	Accelerator	键盘快捷键设置（Ctrl+字符）
ForegroundColor	菜单条上文字的颜色，默认值为黑色	Callback	回调函数，由若干语句组成
Label	菜单上的文字内容	Enable	设置菜单是否有效，默认值为 on
Position	菜单的相对位置序号	Separator	分割行模式，默认值为 off

下面举例说明函数 uimenu 的用法：

```
figure;                                      % 新建图形窗口
H=uimenu(gcf,'label','Tools1');              % 在 Figure 窗口上添加菜单项目 Tools1
H1=uimenu(H,'label','Cut');                  % 在项目 Tools1 中增加 Cut 选项
H2=uimenu(H,'label','Delete');               % 在项目 Tools1 中增加 Delete 选项
H3=uimenu(H,'label','Copy','Separator','on'); % 在项目 Tools1 中增加 Copy 选项
H4=uimenu(H,'label','Undo');                 % 在项目 Tools1 中增加 Undo 选项
H21=uimenu(H2,'label','Curve');              % 在选项 Delete 中增加 Curve 选项
```

运行上述程序，所得菜单展开形式如图 18.2 所示。

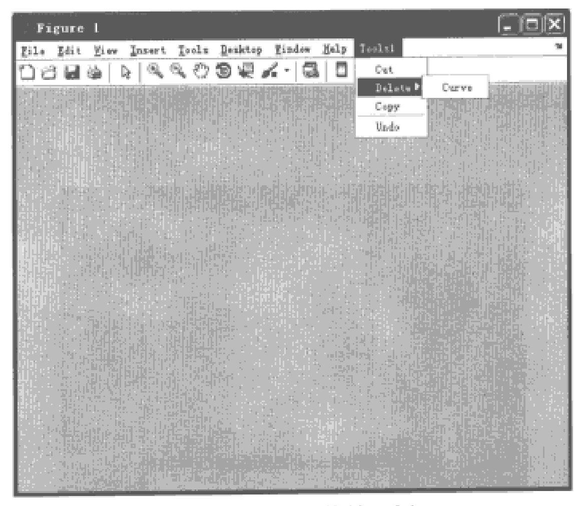


图 18.2　自制菜单示例

函数 uicontextmenu 可以用来创建上下文菜单，其调用格式为：

```
handle = uicontextmenu('PropertyName',PropertyValue,...)
```

参数说明：参数 handle 是函数 uicontextmenu 输出的句柄，通过它用户可以查看和修改上下文菜单的属性值。PropertyName 和 PropertyValue 分别是属性名称和相应的取值。

使用函数 uicontextmenu 一般要结合一些绘图函数，用于创建鼠标在图形对象上单击右键弹出

的菜单。下面以函数 surf 绘制的曲面为例说明函数 uicontextmenu 的用法，相应程序如下：

```
cmenu = uicontextmenu;          % 定义上下文菜单
hline = surf(peaks(30),'FaceColor','none','UIContextMenu', cmenu); %绘制曲面，同时设置上下文菜单
cb1 = ['set(gco, ''LineStyle'', ''--'')'];  % 定义回调函数内容
cb2 = ['set(gco, ''LineStyle'', '':'')'];   % 定义回调函数内容
cb3 = ['set(gco, ''LineStyle'', ''-'')'];   % 定义回调函数内容
cb4 = ['set(gco, ''EdgeColor'', ''y'')'];  % 定义回调函数内容
uimenu(cmenu, 'Label', 'dashed', 'Callback', cb1);  % 利用函数uimenu设置右键弹出菜单
uimenu(cmenu, 'Label', 'dotted', 'Callback', cb2);   % 利用函数uimenu设置右键弹出菜单
uimenu(cmenu, 'Label', 'solid', 'Callback', cb3);    % 利用函数uimenu设置右键弹出菜单
uimenu(cmenu, 'Label', 'yellow', 'Callback', cb4);  % 利用函数uimenu设置右键弹出菜单
```

运行上述程序，用户在曲面对象内右键单击会弹出上下文菜单，如图 18.3 所示。

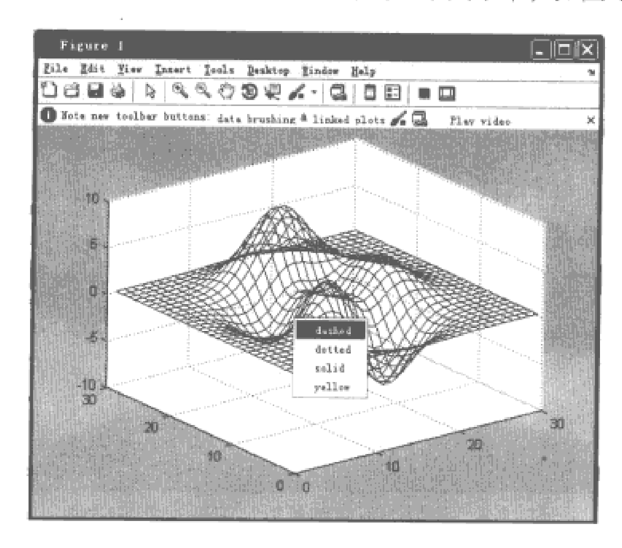


图 18.3 上下文菜单的示例



函数 uicontextmenu 创建对象的句柄作为绘图函数 surf 的 UIContextMenu 属性值，上下文菜单的内容是通过函数 uimenu 建立的，设置相应的回调函数内容可以执行不同的动作，这里设置了 3 种线型和 yellow 等 4 项操作，其中选定的项目将被执行相应回调函数中的内容。

在图 18.3 中依次选择不同的项目可以得到如图 18.4 所示的图形，读者可以比较不同的显示效果。

MATLAB 中部分函数支持 UIContextMenu 属性，如 mesh，surf，plot 等函数，而 fplot 和 ezplot 等函数则不支持这个属性。因此用户在利用 UIContextMenu 属性时，需要事先确定绘图函数是否支持该属性，不支持则需要寻找等价绘图函数来替换。

函数 menu 可生成一个弹出菜单，可供用户单击鼠标选择不同选项，其调用格式为：

```
choice = menu('header', 'item1', 'item2', ... ); '
choice = menu('header', itemlist);
```

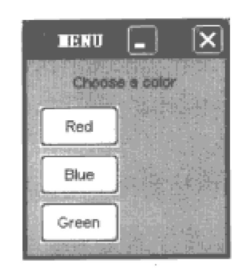
图 18.4　利用上下文菜单控制图形对象的样式

参数说明： choice 是选择结果的序号。Header 表示弹出对话框的提示内容。item1 和 item2 等表示选项按钮上面显示的文字。Itemlist 是由 item1 和 item2 等组成的细胞数组。

下面举例说明函数 menu 的用法。

```
itemlist={'Red','Blue','Green'};                      % 输入颜色字符串
K = menu('Choose a color','Red','Blue','Green')   % 生成菜单控件
K1 = menu('Choose a color again',itemlist)         % 生成菜单控件
```

执行上述程序可以得到如图 18.5 所示的图形窗口。



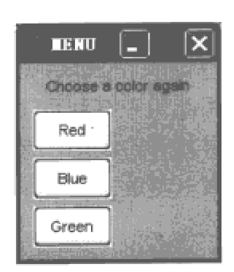


图 18.5　弹出菜单窗口



利用函数 menu 得到的菜单窗口的大小被锁定，用户不能通过鼠标拖动改变大小。窗口默认名称是 MENU。用户单击鼠标选择某个选项之后，这个图形窗口自动关闭。

18.1.2 回调函数设计

回调函数是 GUI 设计的灵魂，控件的响应动作都是由这里定义的。在函数 uimenu 以及回调函数（CallBack）设计中，需要注意的是：回调函数的内容以字符串形式输入，MATLAB 自动执行字符串对应的内容，即相当于 eval('CallBack_str')，因此要求输出的内容无语法错误，回调函数中的内容是合法字符串。值得注意的是，字符串型内容中的单引号需要用两个单引号来替换。比如：

```
a='str';
set(gcf,'Color','w');
```

将上述这样的语句放在回调函数中需要按下述方式替换：

```
uimenu(h1,'label','ABC','CallBack','a=''str'';');                % 在句柄 h1 中添加菜单条
uimenu(h1,'label','DEF','CallBack','set(gcf,''Color'',''w'');');  % 在句柄 h1 中添
加菜单条
```

有的用户把两个单引号看成双引号，从而引起程序错误。

回调函数中，有时输入内容过长，需要进行换行，此时需要把回调函数中的内容分开为几个字符串。不同字符串利用方括号连接起来，在字符串之间用逗号分开，在任意一个逗号后面可以用换行符换行，比如下面的例子：

```
E1=uimenu(gcf,'label','Edit','callback',['a=''Str'';',...
'set(gcf,''Color'',''w'');']);                       % 设置菜单的回调函数
```

在换行符后面不能加入注释内容，需要在整个语句结束后面的位置加入注释文字。

对于比较复杂的回调函数内容，用户可以另行建立 M 文件实现回调函数的内容，然后把 M 文件的内容写入回调函数中即可执行其任务。

18.2 自定义工具条

图 18.6 给出了 Figure 窗口自带工具条功能的说明（其中曲线是利用“plot(0:.01:1,humps(0:.01:1));”得到的），它们辅助绘图，并可以实现一些人机交互操作。

18.2.1 图形编辑功能

由图 18.6 可见，在图形窗口内包含了丰富的图形编辑功能。这里主要介绍激活拖动图形功能按钮、显示数据按钮、选择数据按钮和更新数据曲线按钮等内容。

18.2.1.1 激活拖动图形功能按钮

当用户单击激活拖动图形功能按钮（Pan）后，按下鼠标左键在坐标轴内拖动可以改变坐标轴的显示范围，如图 18.7 所示（鼠标的手形标志未显示在图中）。

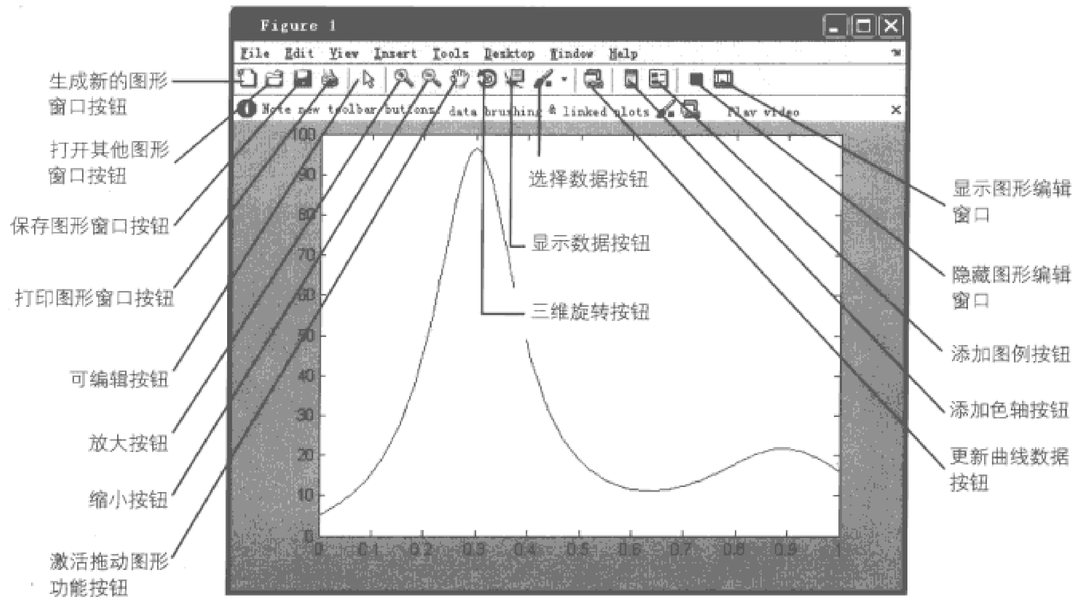


图 18.6　图形窗口按钮功能示意图

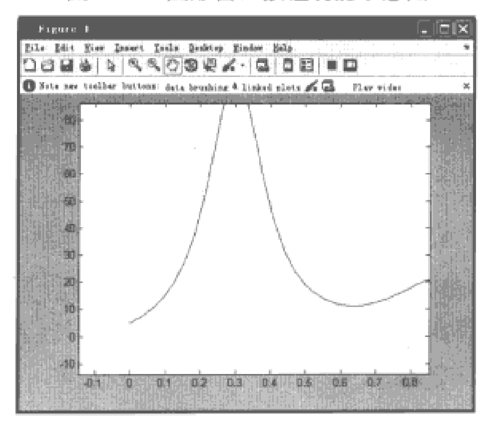


图 18.7　利用鼠标拖动改变坐标轴的显示范围

18.2.1.2　显示数据按钮

当单击显示数据按钮（Data Cursor）后，用户可以在曲线上单击鼠标左键，此时会在曲线上显示该单击点的坐标数值，如图 18.8 所示，这一点在查看曲线数据时非常方便。用户在非曲线区域单击左键不会显示数据。用户单击鼠标右键会显示如图 18.9 所示的菜单。

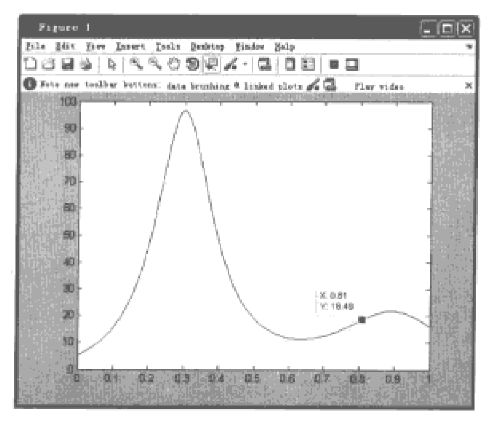


图 18.8 显示曲线上鼠标单击点的数据

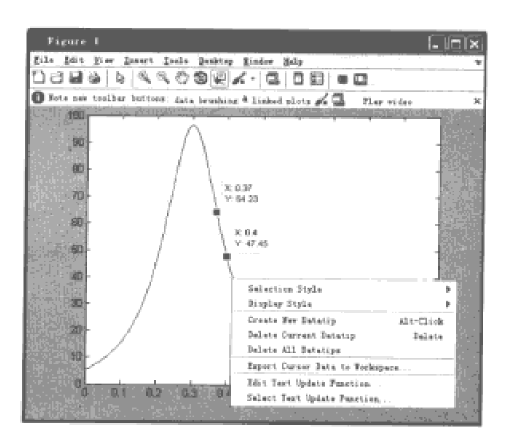


图 18.9 单击鼠标右键显示的菜单

下面介绍各菜单项的意义。

- Selection Style 菜单项用于控制选择数据的方式，其下有两个子菜单项：Mouse Position 获取鼠标所在位置数据，Snap to Nearest Data Vertex 获取最近数据点。
- Display Style 菜单项用来控制数据显示的类型，其下含有两个子菜单项：Window Inside Figure 表示在图形窗口内增加一个窗口显示数据（选择该菜单项，Create New Datatip，Delete Current Datatip，Delete All Datatips 菜单项为灰色，同时窗口内显示的数据被清除），Datatip 以数据标签方式显示（如图 18.8 所示的样式）。
- Create New Datatip 菜单项显示曲线上一个点的数据，如此下去可以显示多个点的数据。
- Delete Current Datatip 菜单项可以删去当前的数据标签。
- Delete All Datatips 菜单项可以删去所有的数据标签。
- Export Data to Workspace 菜单项可以把所选择的数据点输出到工作空间（workspace）中。选择该菜单项时，会弹出一个对话框，如图 18.10 所示，其中用户可以输入变量名然后单击 OK 按钮即可把数据保存到 workspace。被保存的数据是结构体数据。如：

```
cursor_info =
1x3 struct array with fields:
    Target
    Position
    DataIndex
```

其中 Target 对应着曲线的句柄，Position 表示所取点的坐标数据，DataIndex 表示所取点在绘图数据中的位置。



图 18.10 数据存储对话框

- Edit Text Updata Function 菜单项可以编辑文本升级函数，此时会弹出如图 18.11 所示的窗口。用户可以在如图 18.11 所示的窗口中输入自己的语句。
- 选择 Selected Text Updata Function 菜单项时，弹出如图 18.12 所示的对话框用以选择文本

升级函数。

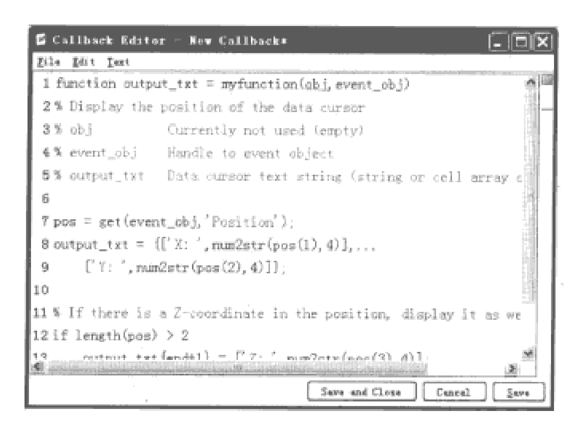


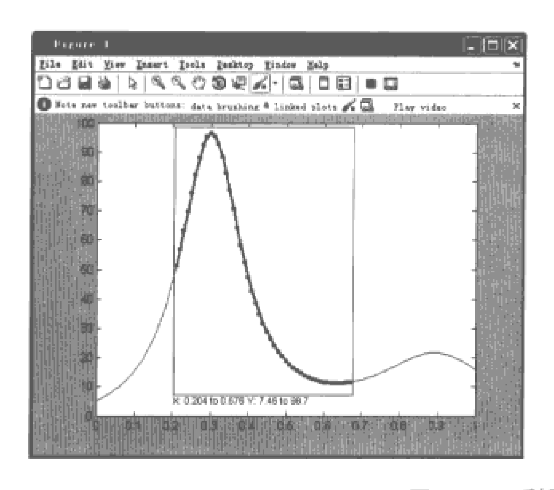
图 18.11 编辑回调函数窗口



图 18.12 选择文本升级函数对话框

18.2.1.3 选择数据按钮

当用户单击选择数据按钮（Brush/ Selected Data）时，用户可以从曲线上选择数据并进行相应的处理。单击这个按钮，用鼠标在曲线上框选出一段数据，如图 18.13 所示的左图。被选择的范围内，绘图数据利用点符号标记，而曲线变为红色，同时线宽增加。在框选过程中显示黑色线框，并在黑色线框下面显示横纵坐标的范围。



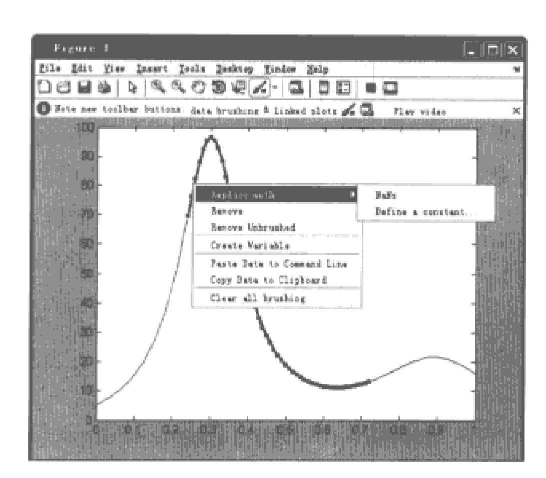
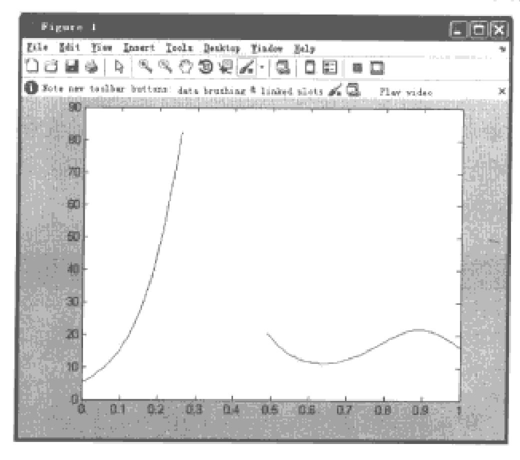


图 18.13 利用鼠标选择数据

用户在选择的曲线上单击鼠标右键后会弹出一个菜单，如图 18.13 所示的右图。用户可以通过菜单来修改曲线。

◆ Replace with 菜单项可以把所选区域的数据利用 NaN 和一个常数替换。NaNs 子菜单项可以把曲线上所选区域删去，Define a constant 子菜单项可以定义一个常数替换曲线的选定数据（此时会弹出如图 18.14 所示的窗口），在编辑框内输入某一常数（默认数值为 0）后单击 OK 按钮就可以进行替换了。利用 NaNs 和 Define a constant 替换的结果如图 18.15 所示。

图 18.14 定义常数的对话框



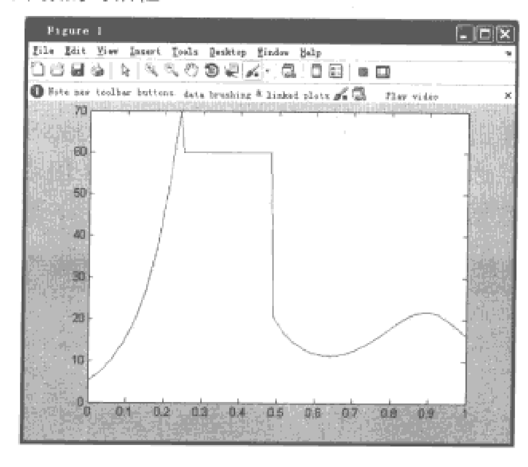


图 18.15 对曲线部分数据进行替换的结果

- Remove 菜单项可以把所选数据删去，剩余的数据再通过绘图命令得到新的图形，如图 18.16 所示。用直线将图 18.15 分开的两段曲线连接起来就得到了图 18.16 所示的结果。
- Remove Unbrushed 菜单项可以把未选定的数据对应的曲线删去，从而保留选定部分的曲线。这个功能可以用于从整体曲线上节选部分曲线。
- Create Variable 菜单项可以把所选曲线对应的数据保存到 workspace 中，选择该菜单项可以弹出一个对话框（如图 18.17 所示），用户在其中输入变量名就可以在 workspace 中得到同名的变量，其为含有两列元素的矩阵。

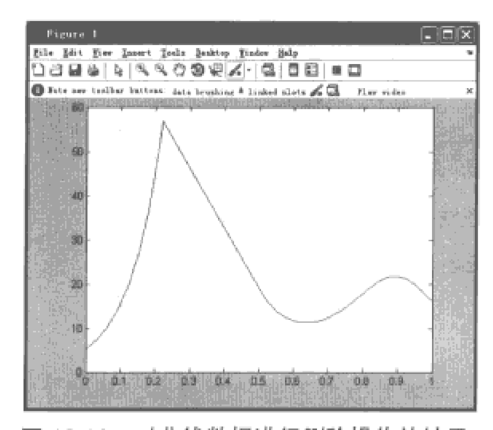


图 18.16 对曲线数据进行删除操作的结果

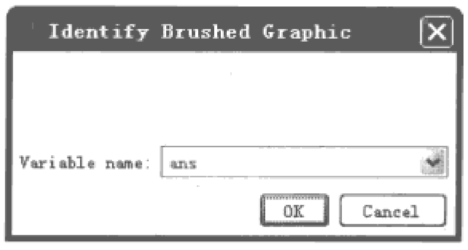


图 18.17 设置变量名

- Paste Data to Command Line 菜单项可以把当前所选线段部分的数据输出到命令窗中，输出的数据是两列元素的矩阵。

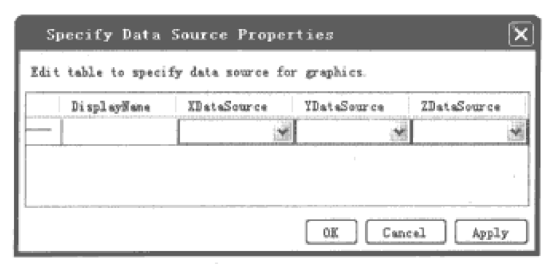
◆ Copy Data to Clipboard 菜单项可以把所选择曲线对应的数据复制到剪贴板，用户在记事本等软件中进行粘贴操作即可得到相应的数据。

◆ Clear all brushing 菜单项用于清除选定的曲线部分的标记，该操作和单击左键效果一样。

18.2.1.4 更新数据曲线按钮

当用户单击更新曲线数据按钮（Link Plot）时，信息提示栏会提示“No graphics have data sources. Cannot link plot: fix it”这样的文字。其中“fix it”对应着一个链接，用鼠标左键单击它会弹出如图 18.18 左图所示的对话框。

单击 XdataSource，YdataSource 和 ZdataSource 标签处的下拉按钮可以显示 workspace 中的变量，选择相应坐标轴的数据后（如图 18.18 右图所示），鼠标单击 OK 或者 Apply 按钮即可把选择的数据绘制到坐标轴上，如图 18.19 所示，而原来的曲线被删除。



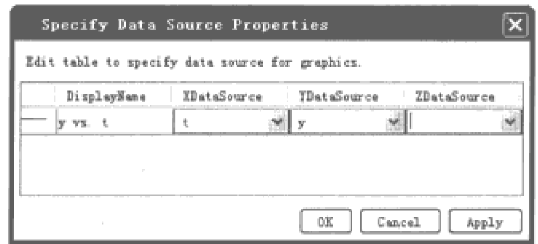


图 18.18 选择数据对话框

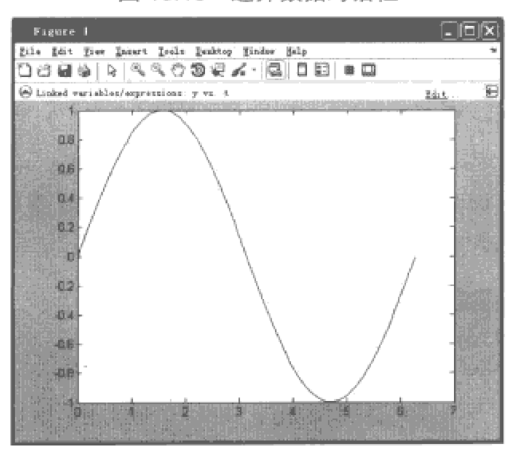


图 18.19 绘图结果

可见利用 MATLAB 提供的这些工具按钮可以方便地进行图形编辑。

18.2.2 个性化图标

除了前面介绍的 MATLAB 提供的工具条外，用户还可以自己设计工具条。相关的函数有 uitoolbar 和 uipushtool，用户利用它们可以在图形窗口内添加自己设计的按钮。下面具体介绍这两个函数的用法。

函数 uitoolbar 可以在图形窗口上增加一个工具条，其调用格式为：

```
TH=uitoolbar(FH, 'PropertyName1', value1, 'PropertyName2', value2,...);
```

参数说明： TH 是工具条输出的句柄。FH 表示图形窗口的句柄。PropertyName1 和 PropertyName2 等表示属性名称。value1 和 value2 等用于指定属性的取值。

函数 uipushtool 可以在工具条中添加按钮，其调用格式为：

```
ph=uipushtool(tbh, 'PropertyName1', value1, 'PropertyName2', value2,...);
```

参数说明： ph 是函数 uipushtool 输出的句柄。tbh 为工具条的句柄，当该句柄缺省时 MATLAB 将在自带工具条下面添加新的按钮。PropertyName1 和 PropertyName2 等表示属性名称。value1 和 value2 等用于指定属性的取值。

18.2.3 参数设置

函数 uipushtool 中的常用属性如表 18.2 所示。

表 18.2 函数 uipushtool 的常用属性说明

属性名称	说明
ClickedCallback	鼠标按下相应函数，相当于回调函数 CallBack，其中用户可以输入程序段
CData	按钮图案数据，为一个真彩色数据矩阵，要求数组的 size 为 M×N×3
Enable	控制菜单是否有效，默认值为 on
Separator	控制按钮之间是否分隔，默认值为 off
TooltipString	鼠标指针位于该按钮时显示的提示信息

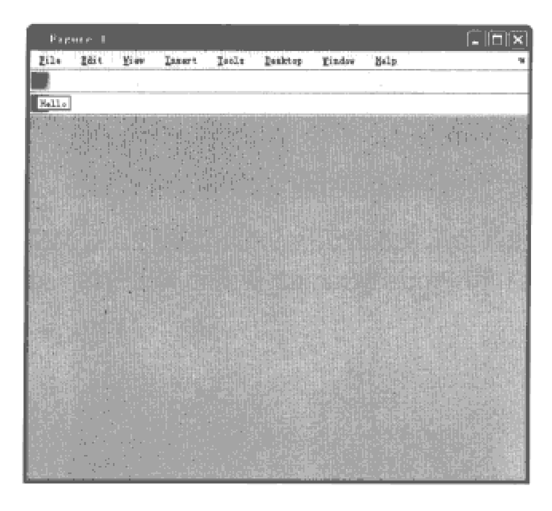
下面举例说明利用函数 uitoolbar 和 uipushtool 制作按钮，程序如下：

```
h = figure('ToolBar','none');       % 设置图形窗口自带按钮为隐藏模式
ht1 = uitoolbar(h);                 % 生成工具条 1
ht2 = uitoolbar(h);                 % 生成工具条 2
a = [.05:.05:0.95];                 % 生成离散数据
b(:,:,1) = repmat(a,19,1)';         % 生成按钮图案真彩色数组
b(:,:,2) = repmat(a,19,1);          % 生成按钮图案真彩色数组
b(:,:,3) = repmat(flipdim(a,2),19,1);% 生成按钮图案真彩色数组
hpt1 =
uipushtool(ht1,'CData',b,'TooltipString','Hello','ClickedCallback','why');%在工
具条 1 中添加按钮
hpt2 =
uipushtool(ht2,'CData',rand(20,20,3),'TooltipString','clc','ClickedCallback','
clc');
```

上述程序运行后可以得到如图 18.20 所示的结果。



函数 uitoolbar 用于在图形窗口内产生一排工具条，在其中可以添加按钮。本例中共产生两排按钮。在图 18.21 中，用户单击渐变颜色的按钮，可以显示随机的英文（函数 why 的输出结果），单击随机颜色的按钮可以执行清屏操作。



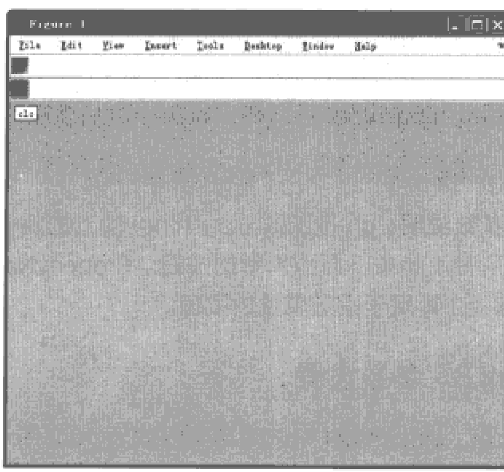


图 18.20　自制按钮的示例

此外，函数 uitoggletool 还可以用来创建开关按钮，其用法和函数 uipushtool 类似，用户可以查阅帮助信息了解它的使用。用户模仿前面给出的例子可以产生自己需要的按钮。在属性 ClickedCallback 中加入相应的程序段可以实现一定的功能。

18.3　控件设计

除了前面介绍的菜单条和按钮的设计外，在图形窗口内，用户还可以建立按钮（pushbutton）、开关按钮（togglebutton）、无线按钮（radiobutton）、检查框（chechbox）、编辑框（edit）、文本框（text）、滑动条（slider）、空白框（frame）、列表框（listbox）和下拉菜单（popupmenu）等控件形式。这些控件可以通过 GUI 设计工具或者 MATLAB 函数来生成。本节介绍利用函数制作这些按钮，关于利用 GUI 设计工具制作控件的方法将在后面的章节介绍。

18.3.1　基本函数

MATLAB 提供了函数 uicontrol 制作前面介绍的按钮、滑动条、文本框及弹出式下拉菜单等控件。函数 uicontrol 的调用格式如下：

```
H1=uicontrol(Hf, 'PropertyName', PropertyValue, ...);
```

参数说明：H1 是制作的控件对应的句柄。Hf 是图形窗口的句柄。PropertyName 和 PropertyValue 表示控件的属性和相应的取值。常用的属性名称和说明如表 18.3 所示。

表 18.3　函数 uicontrol 生成的控件的属性说明

属性名称	说明
BackgroundColor	设置控件的背景颜色，其默认值与系统的窗口属性设置有关，用户可以设定其为特殊颜色控制符或者 3 个元素组成的向量
Callback	输入回调函数内容，其由合法的程序段组成
CData	显示在控件表面的真彩色图案对应的数组，其为 M×N×3 的数组
Enable	控制控件是否有效，默认值为 on
FontAngle	控制文本字符是否倾斜，默认值为 normal

（续表）

属性名称	说明
FontName	字体名称，默认字体为 MATLAB 系统设定的字体
FontSize	字体大小，默认字体与 MATLAB 设置有关
FontUnits	字体大小的度量单位，默认值为 points。其中 normalized 选项是归一化坐标，可使字体在缩放图形窗口时保持字体大小相对不变
FontWeight	字体的磅值，默认值为 normal
ForegroundColor	控件上文本的颜色，其默认值与系统的窗口属性设置有关
HorizontalAlignment	控件上文本水平对齐方式，默认值为 center（中心对齐）
ListboxTop	显示于列表框中项目的索引
Max	最大值，对部分控件有效
Min	最小值，对部分控件有效
Position	设置控件对象的位置与大小，其值为 4 个元素组成的向量（元素取值是相对于图形窗口而言的）
String	显示在控件对象上的标识文本，其值为字符型数据
Style	设置创建控件对象的类型，默认值为 pushbutton
SliderStep	滑块的步长和尺度，默认值为[0.01, 0.1]，其中第一个元素是步长，第二个元素表示滑块的大小
TooltipString	控件对象的提示信息，即鼠标放置于控件上显示的文字，其值为任意字符
Units	控件对象显示时对应的度量单位，默认值为 pixels
Value	控件的当前值，该属性对部分控件有效

函数 uicontrol 是一个通用函数，利用这个函数结合前面表格中介绍的属性就可以制作不同类型的控件。所得控件被放置在图形窗口的绘图区域中。

18.3.2 控件基础

下面调用函数 uicontrol 制作不同类型的控件，程序如下：

```
ed=uicontrol(gcf,'style','pushbutton','unit','normalized','position',[0.1,0.9,
0.1,0.06],'fontsize',12,'String','按钮');
to=uicontrol(gcf,'style','togglebutton','unit','normalized','position',[0.1,0.
8,0.1,0.06],'fontsize',12,'String','开关按钮');
ra=uicontrol(gcf,'style','radiobutton','unit','normalized','position',[0.1,0.7
,0.12,0.06],'fontsize',12,'String','无线按钮');
ch=uicontrol(gcf,'style','checkbox','unit','normalized','position',[0.1,0.6,0.
12,0.06],'fontsize',12,'String','无线按钮');
ed=uicontrol(gcf,'style','edit','unit','normalized','position',[0.1,0.5,0.12,0
.06],'fontsize',12,'String','编辑框');
te=uicontrol(gcf,'style','text','unit','normalized','position',[0.1,0.4,0.12,0
.06],'fontsize',12,'String','文本框');
sl=uicontrol(gcf,'style','slider','unit','normalized','position',[0.1,0.3,0.2,
0.06],'fontsize',12,'String','滑动条');
fr=uicontrol(gcf,'style','frame','unit','normalized','position',[0.3,0.6,0.4,0
.3],'fontsize',12,'String','空白框');
li=uicontrol(gcf,'style','listbox','unit','normalized','position',[0.4,0.2,0.2
,0.3],'fontsize',12,'String','列表框|B0|C');
po=uicontrol(gcf,'Style','popupmenu','unit','normalized','position',[0.7,0.2,0
.2,0.3],'fontsize',12,'String','弹出菜单|A|B|C');
```

利用函数 uicontrol 可以得到不同类型的控件，控件的分类可以使用其中的属性 Style 设置。还

可以设置位置（Position）、字体（Fontsize）以及显示字符（String）等属性。上述程序执行后得到的图形如图 18.21 所示。

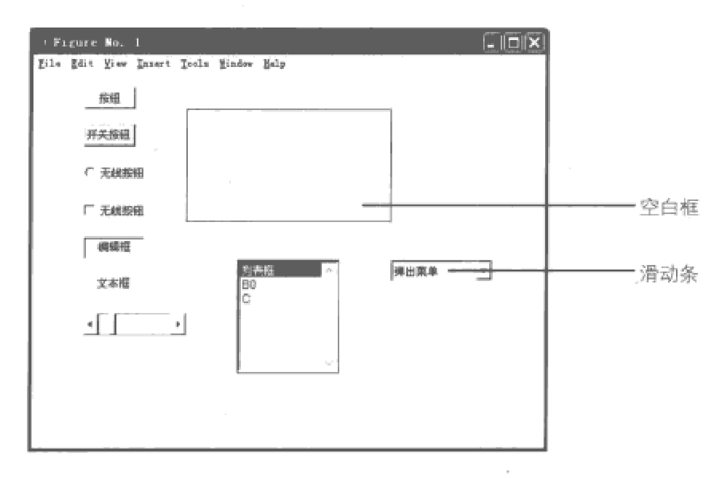


图 18.21　全部控件的外观图



除了滑动条和空白框外，其他控件可以通过属性 String 添加标注文字信息。而空白框控件可以用于 GUI 设计中分隔区域。图 18.22 中只是给出了控件的外观，但关于回调函数、字体等属性未设定。

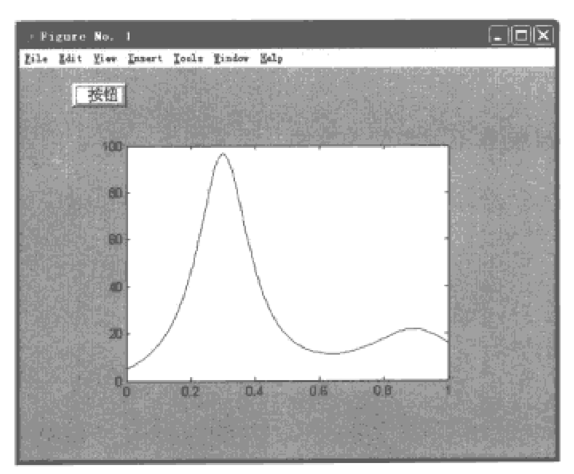


图 18.22　按钮空间示例

18.3.3　回调函数设计

文本框和空白框控件用于生成说明性标注，它们的回调函数属性很少使用。本小节介绍除文本框控件和空白框控件外的几种控件的回调函数设计。

18.3.3.1　按钮控件

对于按钮控件，单击按钮时，回调函数（Callback）部分内容将被执行。把需要执行的程序内容输入到 Callback 属性中即可。比如下面的程序段：

```
figure;  % 新建图形窗口
uicontrol(gcf,'style','pushbutton','unit','normalized','position',[0.1,0.9,0.1
,0.06],'fontsize',12,...
 'String','按钮
','Callback','axes(''Position'',[0.2,0.2,0.6,0.6]);fplot(@humps,[0,1]);');  %
生成按钮
```

当执行上述程序，并单击按钮时可以得到如图 18.22 所示的图形。



对于按钮属性中 Style 的输入内容，输入 push 程序也可运行。

18.3.3.2 开关按钮

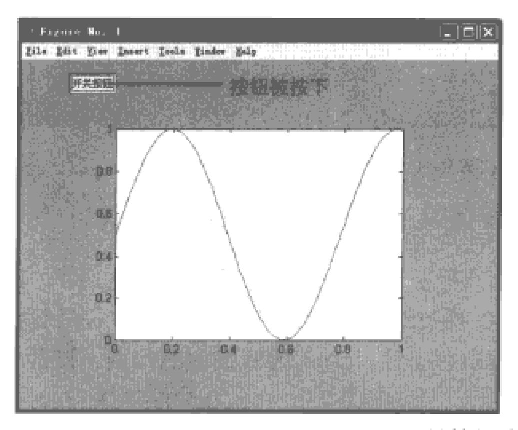
对于开关按钮，当按钮被按下时属性 Value 的值等于 1，而当还原按钮时 Value 的值等于 0。利用这个按钮可以控制属性 Value 选取两个不同的数值。通过语句“v=get(to,'value');”可以获取属性 Value 的取值（其中 to 是开关按钮的句柄）。下面举例说明开关按钮的制作，程序如下：

```
figure;               % 新建图形窗口
t=linspace(0,1);        % 生成离散采样数据
to=uicontrol(gcf,'style','togglebutton','unit','normalized','position',[0.1,0.
9,0.1,0.06],'fontsize',12,...
   'String','开关按钮
','Callback',[ 'v=get(to,''value'');axes(''Position'',[0.2,0.2,0.6,0.6]);',...
      'if v>0.5; ph=plot(t,[1+sin(t*8)]/2,''r'');elseif v<0.5 delete(ph);box
on;end;' ]);
```



Callback 属性值中是两个单引号，而非双引号（后面类似情况也如此）。其中利用 if 语句判断属性 Value 的不同取值而采取不同的动作。

当用户把开关按钮置于不同状态时可以得到如图 18.23 所示的两个图形。



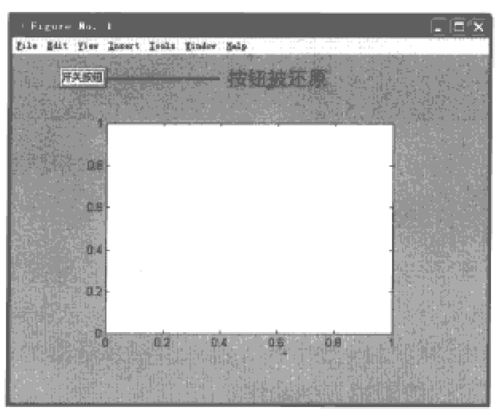


图 18.23 开关按钮处于两种状态时的图形

18.3.3.3 无线按钮

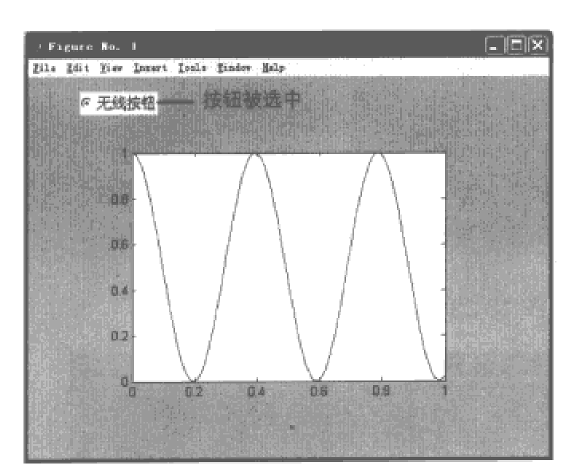
无线按钮和开关按钮功能相似，当无线按钮前面的圆圈被选中（圆圈中有黑点）时，其属性 Value 的值等于 1，未被选中时属性 Value 的值等于 0。通过选中与未选中两种状态可以得到两个值。其中利用语句“v=get(ra,'value');”可以获得无线按钮的状态值，ra 是无线按钮的句柄。下面举例说明无线按钮的制作与使用，程序如下：

```
figure;                  % 新建图形窗口
t=linspace(0,1);         % 生成离散采样数据
uicontrol(gcf,'style','radiobutton','unit','normalized','position',[0.1,0.9,0.
15,0.06],'fontsize',12,...
    'String','无线按钮
','Callback',[ 'v=get(gco,''value'');axes(''Position'',[0.2,0.2,0.6,0.6]);',...
    'if v>0.5; ph=plot(t,[1+cos(t*16)]/2,''b'');elseif v<0.5 delete(ph);box on;end;
' ]);  % 生成无线按钮
```



其中 gco 表示当前对象的句柄（下面类似情况中的 gco 也是如此），在无线按钮被鼠标单击时，它就是当前对象，即其句柄可以用 gco 表示。这样函数 uicontrol 就不用输出句柄了。

执行上述程序并置无线按钮为两种不同状态，则有如图 18.24 所示的结果。



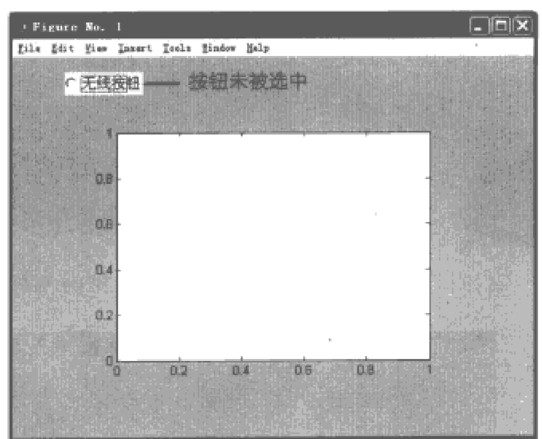


图 18.24　无线按钮处于两种状态时的图形

18.3.3.4 检查框

检查框与无线按钮样式相似，当检查框前面的方框被选中（方框中有对钩符号）时，其属性 Value 的值等于 1，未被选中时属性 Value 的值等于 0。通过选中与未选中两种状态可以得到两个值。其中利用语句“v=get(ch,'value');”可以获得检查框的状态值，ch 是无线按钮的句柄，根据 v 的不同数值可以做出相应的动作。

下面举例说明检查框的制作与使用，程序如下：

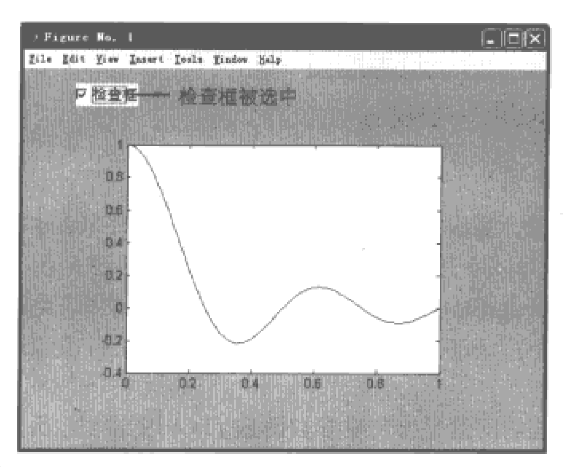
```
figure;                  % 新建图形窗口
t=linspace(0,1);         % 生成离散采样数据
```

```
uicontrol(gcf,'style','checkbox','unit','normalized','position',[0.1,0.9,0.12,
0.06],'fontsize',12,...
    'String','检查框
','Callback',['v=get(gco,''value'');set(gca,''Position'',[0.2,0.2,0.6,0.6]);',
...
        'if v>0.5;ph=plot(t,sinc(t*4));elseif
v<0.5;set(ph,''Visible'',''off'');end;']);     % 生成检查框
```



这里使用曲线是否显示来举例说明检查框的制作和使用，用户还可以替换为其他响应动作。

上述程序所得的效果如图 18.25 所示。



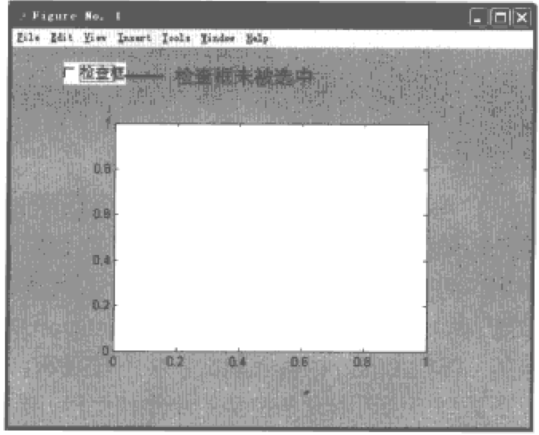


图 18.25 检查框处于两种状态时的图形

18.3.3.5 编辑框控件

对于编辑框控件，用户可以在其中输入相应的内容并被 MATLAB 获知，通过回调函数可以把从编辑框中获得的数据作为下一步计算的输入参数。获取编辑框中的内容可以通过语句“str=get(ed,'string');”获得，其中 ed 是编辑框的句柄。此语句需要写入到编辑框控件的回调函数中。下面举例说明编辑框控件的制作和使用，程序如下：

```
figure;                    % 新建图形窗口
uicontrol(gcf,'style','edit','unit','normalized','position',[0.1,0.9,0.42,0.06
],'fontsize',12,...
    'String','这是编辑框，请输入一个合法数值
','Callback',['str=get(gco,''string'');',...
        'set(gca,''Position'',[0.2,0.2,0.6,0.6]);cla;text(0.3,0.6,''你输入的数值
是:'');',...
        'text(0.3,0.5,str,''Fontsize'',14);']);      % 生成编辑框
```



从编辑框句柄属性 String 得到的内容是字符串型数据，用户可以使用函数 str2num 把字符串型数据转化为数值型数据。

执行上述程序，并在编辑框中输入“2222”后按 Enter 键后得到如图 18.26 所示的图形。

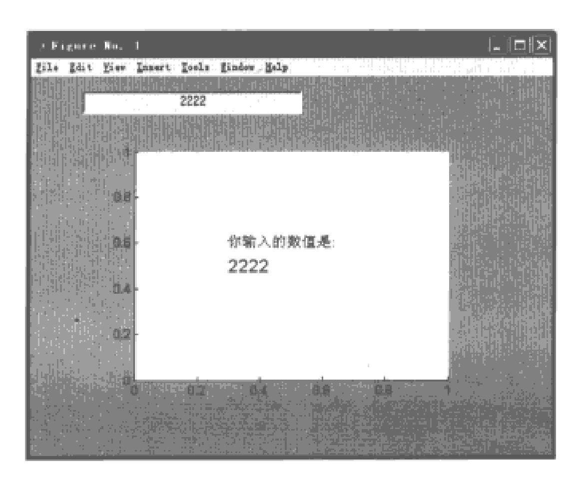


图 18.26　编辑框控件的制作与使用示例

18.3.3.6　滑动条控件

对于滑动条控件，当上面的滑块处于不同位置时，对应的属性 Value 取值不同。属性 Value 的取值范围是[0, 1]，用户如果需要其他取值范围时，可以通过线性变换（A*[0, 1]+B）把区间[0, 1]映射到区间[B,B+A]。滑块的步长和大小利用属性 SliderStep 来设定。利用鼠标可以拖动滑块到不同位置，滑块的位置可以通过语句“v=get(gco,'value');”来获得，返回值 v 是在 0~1 之间的 double 型数据。属性 String 内输入的文字将不被显示。

下面举例说明滑动条控件的制作和使用，要处理的问题是绘制变参数函数 $f(x,v)$ 的曲线。函数 $f(x,v)$ 的表达式为 $f(x,v)=\cos(3t)\text{sinc}(vt)$，其中参数 v 的取值范围是[3, 5]，通过滑动条控件来显示不同参数下的函数曲线情况。

分析：滑动条正好可以通过滑动滑块来得到不同的取值，滑动条控件输出参数的范围是[0, 1]，而本问题中参数的取值范围是[3, 5]，因此需要通过前面介绍的表达式(A*[0, 1]+B)来转化，A=2，B=3。这里滑动条的步长取为 0.005，由此滑块移动的单位长度就是 0.01。同时设计属性 TooltipString 来实时显示参数 v 的取值。

根据前面的分析可以写出相应的程序，程序内容如下：

```
figure;                                  % 新建图形窗口
set(gca,'Position',[0.2,0.2,0.6,0.6]);  % 设置当前坐标轴的位置
ezplot('cos(3*t)*sinc(4*t)',[0,pi]);    % 绘制默认状态下曲线图
uicontrol(gcf,'style','slider','unit','normalized','position',[0.1,0.9,0.8,0.0
4],'fontsize',12,...

'SliderStep',[0.005,0.01],'Value',0.5,'Callback',['v=get(gco,''value'');',...
       'ezplot([''cos(3*t)*sinc('',num2str(2*v+3),''*t)''],[0,pi]);',...
       'set(gco,''TooltipString'',[''v='',num2str(2*v+3)]);']);  % 生成滑动条
```

执行上述程序可以得到如图 18.27 所示的图形。

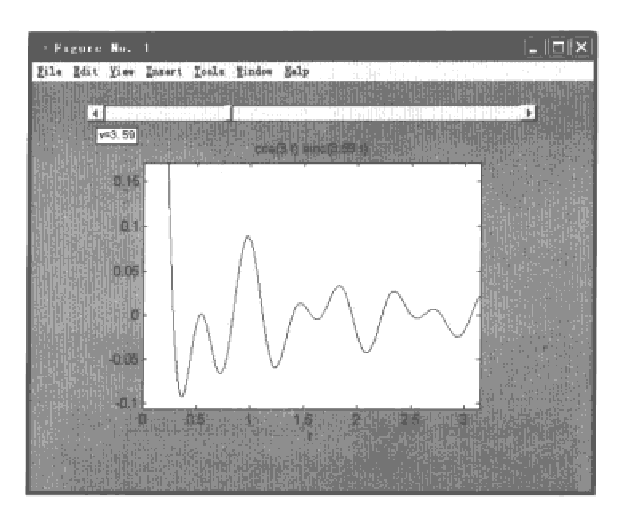


图 18.27 滑动条控件制作和使用的示例



在图 18.27 所示的界面中，用户单击滑动条两端带黑色三角块的按钮可以增加（对应右侧按钮）或者减小（对应左侧按钮）参数 v 的数值，曲线随着参数 v 的变化而变化，这时参数 v 变化的步长为 0.01。此外用户单击或者按住滑动条中间非滑块区域来增减参数 v 的数值（这时参数 v 变化的最小步长为 0.02），当用户按住滑块拖动时可以得到带有 3 位小数的 v 的数值。

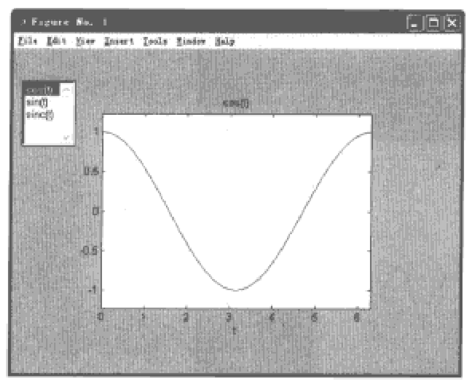
18.3.3.7 列表框

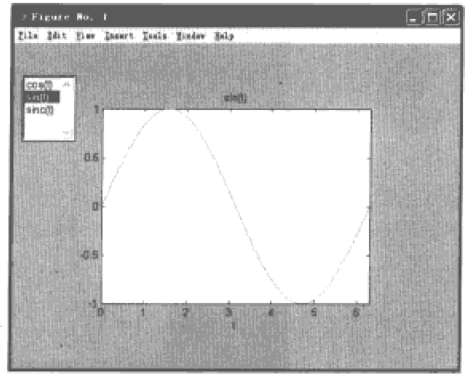
对于列表框可以同时显示多个选项供用户选择，各个选项的名称在属性 String 中输入，不同选项名称利用符号“|”来分隔。不同选项的区分是通过属性 Value 的数值来体现的。Value 的取值对应于选项自上而下的序号，用户可以设定不同选项对应不同的执行动作。下面举例说明列表框的制作和使用，程序内容如下：

```
figure;                              % 新建图形窗口
set(gca,'Position',[0.2,0.2,0.6,0.6]);  % 设置当前坐标轴的位置
ezplot('cos(t)',[0,2*pi]);                % 绘制默认状态下曲线图
str={'cos(t)','sin(t)','sinc(t)'};          % 设置细胞数组存储不同的函数表达式
col={'r','g','b'};                       % 设置细胞数组存储不同的颜色
uicontrol(gcf,'style','listbox','unit','normalized','position',[0.02,0.7,0.12,
0.2],'fontsize',12,...
    'String','cos(t)|sin(t)|sinc(t)','Callback',['v=get(gco,''value'');',...

'ss=[''ezplot('''''',char(str(v)),'''''',[0,pi*2,-1,1])'']; eval(ss);',...
        'set(get(gca,''Children''),''Color'',char(col(v)));']);  % 生成列表框
```

执行上述程序，分别选择不同选项可以得到如图 18.28 所示的 3 个不同图形窗口。





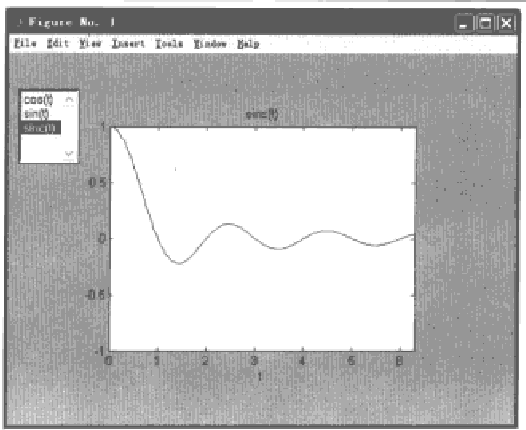


图 18.28　选择列表框中不同选项得到的图形



制作列表框控件的程序中所有引号都是单引号，用户在使用时需要注意。这里再介绍一种等价的方法来替换这样冗长的回调函数内容。可以把回调函数内的程序内容写为一个函数文件形式（脚本文件亦可），相应程序内容为：

```
function draw_all(v);
col='rgb';                          % 存储颜色的变量
if v==1;
   ezplot('cos(t)',[0,pi*2,-1,1]);    % 绘制余弦函数
elseif v==2;
   ezplot('cos(t)',[0,pi*2,-1,1]);    % 绘制正弦函数
elseif v==3;
   ezplot('sinc(t)',[0,pi*2,-1,1]);    % 绘制 sinc 函数
end
set(get(gca,'Children'),'Color',col(v)); % 设置曲线颜色
```

上面程序中可以根据 v 的取值绘制不同样式的曲线。列表框制作程序变为：

```
figure;                                 % 新建图形窗口
set(gca,'Position',[0.2,0.2,0.6,0.6]);     % 设置当前坐标轴的位置
ezplot('cos(t)',[0,2*pi]);                % 绘制默认状态下的曲线图
uicontrol(gcf,'style','listbox','unit','normalized','position',[0.02,0.7,0.12,
0.2],'fontsize',12,...
```

```
'String','cos(t)|sin(t)|sinc(t)','Callback',['v=get(gco,''value'');draw_all(v)
;']);  % 生成列表框
```

可见此时回调函数中的内容变得简单，仅为获取列表框选项序号语句和调用函数文件 draw_all 的语句，用户不必考虑过多引号设计问题。等价形式的程序所得结果与图 18.28 相同。

18.3.3.8 下拉菜单

下拉菜单控件的制作和列表框相似，下面举例说明。

```
figure;                              % 新建图形窗口
set(gca,'Position',[0.2,0.2,0.6,0.6]);   % 设置当前坐标轴的位置
ezplot('cos(t)',[0,2*pi]);                  % 绘制默认状态下曲线图
str={'cos(t)','sin(t)','sinc(t)'};              % 设置细胞数组存储不同的函数表达式
col={'r','g','b'};                          % 设置细胞数组存储不同的颜色
uicontrol(gcf,'style','popupmenu','unit','normalized','position',[0.02,0.7,0.1
2,0.2],'fontsize',12,...
    'String','cos(t)|sin(t)|sinc(t)','Callback',['v=get(gco,''value'');',...

'ss=[''ezplot('''''',char(str(v)),'''''',[0,pi*2,-1,1])''];eval(ss);',...
        'set(get(gca,''Children''),''Color'',char(col(v)));']);  % 生成下拉菜单
```

上述程序中只是在列表框制作例子的基础上，把“listbox”改为“popupmenu”。执行上述程序并单击下拉菜单即可得到如图 18.29 所示的图形。选择不同选项可以得到不同的曲线。

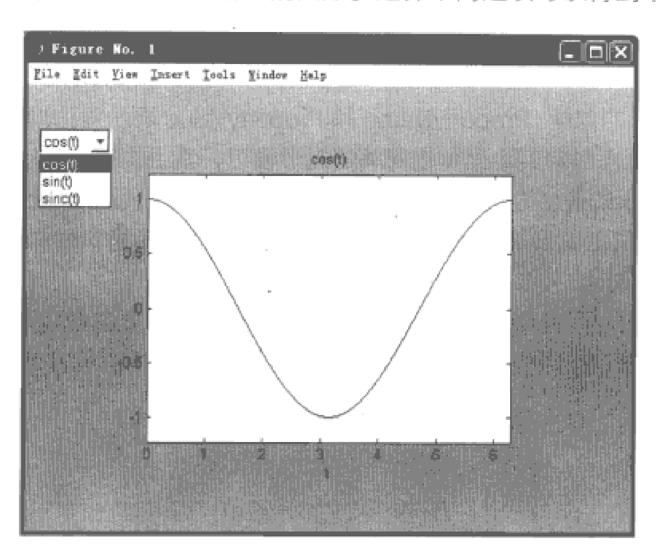


图 18.29 下拉菜单的制作示例



与列表框控件相比，下拉菜单可以节省在图形窗口上的占用面积，但直观性降低。

利用本节介绍的控件回调函数的设计方法就可以制作实现具有一定功能的用户界面窗口了。在 MATLAB 中不同控件的生成程序相似，不同控件的属性含义和赋值方法大体相同，所以用户复制生成控件的程序并更改属性 Style 和 Position 等属性值就可以在图形窗口中得到不同控件。

18.4 对话框

本节介绍 MATLAB 提供制作对话框的函数，如对话框、错误对话框、帮助对话框等，相应的实现函数如表 18.4 所示。

表 18.4 创建不同类型对话框的函数

函数名称	功能	函数名称	功能
dialog	对话框	questdlg	问题对话框
errordlg	错误对话框	uigetfile	文件检索对话框
helpdlg	帮助对话框	uiputfile	写入文件时显示的检索对话框
inputdlg	输入对话框	uisetcolor	设置颜色的对话框
listdlg	选择列表对话框	uisetfont	设置字体的对话框
msgbox	消息对话框	warndlg	警告对话框
pagesetupdlg	显示页面的版面对话框	waitbar	显示程序计算进度的窗口
printdlg	显示打印对话框		

下面依次介绍这些函数的用法。

18.4.1 图形窗口

函数 dialog 生成一个空白的图形窗口，上面没有菜单、工具条和图名。函数 dialog 的调用格式为：

```
h = dialog('PropertyName',PropertyValue,...);
```

参数说明： h 是返回的句柄。PropertyName 和 PropertyValue 表示对话框的属性名称和属性值。函数 dialog 创建的对话框对象实际上是最简单的图形窗口，其上内容为空。对话框的属性和图形窗口属性相同。需要指出的是创建的对话框的属性 WindowStyle 被设为 modal，即典型窗口，此时用户需要用鼠标在该窗口上做出响应后才能进行其他操作。而图形窗口的属性 WindowStyle 的默认值为 normal（标准窗口）。

执行语句 “dh=dialog;” 得到的图形如图 18.30 所示。用户需要在窗口上的灰色空白区域内单击左键或者单击右上角的关闭按钮关闭对话框，才能在 MATLAB 中进行其他操作。

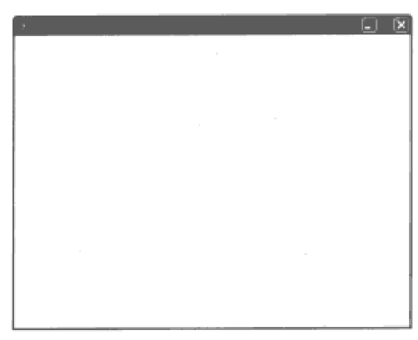


图 18.30 函数 dialog 创建的图形窗口

18.4.2 错误对话框

函数 errordlg 可以用来创建错误对话框，其调用格式为：

```
handle = errordlg('errorstring', 'dlgname', 'createmode');
```

参数说明： handle 是错误对话框的句柄。errorstring 是在对话框上显示的错误信息。dlgname 指定对话框的标题。createmode 用于前端显示控制，表示如果对话框已存在，参数 on 将对话框显示在最前端，参数 off 不把对话框显示在最前端。

执行语句“handle=errordlg('The input value is wrong','Error!!');”可以得到如图 18.31 所示的对话框。

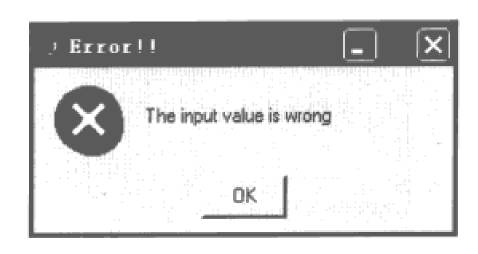


图 18.31 错误对话框

18.4.3 帮助对话框

函数 helpdlg 可以用来创建帮助对话框，其调用格式为：

```
handle = helpdlg('helpstring','dlgname');
```

参数说明： handle 是输出的句柄。helpstring 是显示的帮助信息。dlgname 是帮助对话框的名称。

执行语句“h = helpdlg('This is a help string','My Help Dialog');”可以得到如图 18.32 所示的对话框。

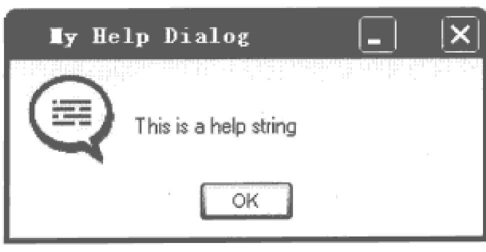


图 18.32 帮助对话框

18.4.4 输入对话框

函数 inputdlg 可以创建输入对话框，其调用格式为：

```
answer=inputdlg(prompt,name,numlines,defaultanswer);
```

参数说明： answer 是输出的结果，其为一个细胞数组。prompt 是一个细胞数组，表示输入选

项的提示信息。name 是输入对话框的名称。numlines 表示输入部分的行数。defaultanswer 用于设置默认输入值。

下面举例说明函数 inputdlg 的用法。

```
prompt={'Enter the matrix size for x^2:','Enter the colormap name:'};   % 定义项目提示信息
name='Input for Peaks function';     % 定义对话框名称
numlines=2;                     % 定义输入区行数
defaultanswer={'20','hsv'};          % 定义默认值
answer=inputdlg(prompt,name,numlines,defaultanswer)              %生成输入对话框
```

执行上述程序将会弹出如图 18.33 所示的对话框。

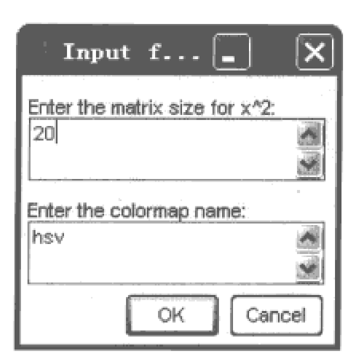


图 18.33　输入对话框

单击图 18.32 所示的 OK 按钮将会在命令窗中出现：

```
answer =
    '20'
    'hsv'
```

18.4.5　列表对话框

函数 listdlg 可用于创建选择列表对话框，其调用格式为：

```
[SELECTION,OK] = listdlg('ListString',S,'PropertyName1',Value1,...);
```

参数说明： SELECTION 表示选择项目的序号。OK 表示选择是否成功，当 OK=1 时表示选择成功，当 OK=0 时表示选择失败，此时 SELECTION 是空数组。ListString 设定选择内容的属性名称。S 是相应的细胞数组。PropertyName1 和 Value1 是属性名称和相应取值。除 ListString 外的属性如表 18.5 所示。

表 18.5　输入对话框的属性

属性名称	说明	属性名称	说明
SelectionMode	设置选择模式：单选和多选（默认值）	PromptString	提示信息对应的字符串
ListSize	列表区域的大小，单位为像素	OkString	设置 OK 按钮的标注文本
InitialValue	设置最初的选择	CancelString	设置 Cancel 按钮的标注文本
Name	设置对话框名称，默认值为空字符串		

下面举例说明 listdlg 的用法。

```
d = dir;str = {d.name};  % 获得当前路径内的文件内容并提取文件名记录为 str
[s,v] = listdlg('PromptString','Select a file:',...

'SelectionMode','single','ListString',str,'InitialValue',3,'ListSize',[160,100
])
```

上述程序执行后得到如图 18.34 所示的图形。

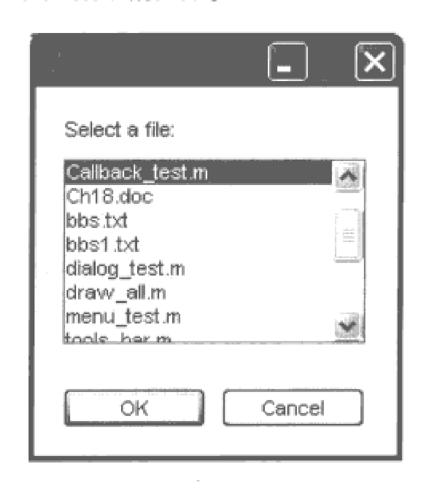


图 18.34　选择列表对话框

18.4.6　消息对话框

函数 msgbox 可以创建消息对话框，其调用格式为：

```
h=msgbox('Message')
h=msgbox('Message','Title')
h=msgbox('Message','Title','Icon')
h=msgbox(Message,Title,' Icon',IconData,IconCMap)
```

参数说明： h 是消息对话框的句柄。Message 是提示消息的内容。Title 是消息对话框的名称。Icon 用于指定图标类型，其可选参数为：none 表示空缺，error 表示使用错误对话框的图标，help 表示使用帮助对话框的图标，warn 表示使用警告对话框的图标，custom 表示用户自定义图标。IconData 是图标图案的数据，其为真彩色图像对应的数组。IconCMap 颜色影像数组，其为含 3 列元素的数组。

下面举例说明消息对话框的制作。

```
Data=1:64;Data=(Data'*Data)/64;              % 生成图标图案的数据
h=msgbox('String','Title','custom',Data,hot(64))  % 生成消息对话框
```

执行上述程序，所得对话框如图 18.35 所示。

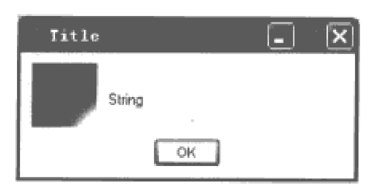


图 18.35　消息对话框

18.4.7　版面对话框

函数 pagesetupdlg 可以显示页面的版面对话框，其调用格式为：

```
dlg = pagesetupdlg(fig);
```

参数说明： dlg 表示版面对话框的句柄。fig 是图形窗口的句柄。

执行语句“dlg = pagesetupdlg(gcf);”可以得到如图 18.36 所示的对话框。

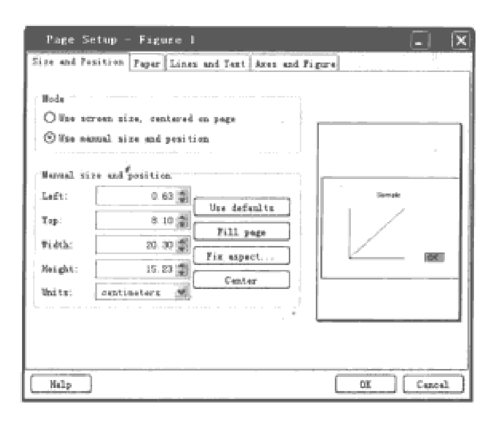


图 18.36　显示页面的版面对话框

18.4.8　打印对话框

函数 printdlg 可以显示打印对话框，其调用格式为：

```
printdlg(fig)
```

参数说明： fig 表示目标图形窗口的句柄。

执行语句“printdlg(gcf);”可以得到如图 18.37 所示的打印对话框。

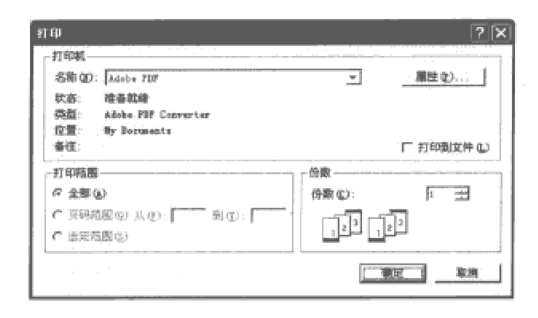


图 18.37　打印对话框

18.4.9　问题对话框

函数 questdlg 可以创建问题对话框，其调用格式为：

```
ButtonName = questdlg('Question', 'Title', 'Btn1', 'Default')
ButtonName = questdlg('Question', 'Title', 'Btn1', 'Btn2', 'Default')
ButtonName = questdlg('Question', 'Title', 'Btn1', 'Btn2', 'Btn3', 'Default')
```

参数说明：ButtonName 返回按下按钮的名称。Question 显示提示问题。Title 是问题对话框的名称。Btn1，Btn2 和 Btn3 表示按钮的名称。Default 表示虚线框标记的按钮名称，其内容必须与前面的按钮名称相同，否则 MATLAB 会忽略最后一个字符串。

下面举例说明函数 questdlg 的用法。

```
ButtonName = questdlg('What is your favorite color?', 'Color Question','Red',
'Green', 'Blue', 'Blue1')
```

执行上述语句可以得到如图 18.38 所示的对话框。



图 18.38　问题对话框

18.4.10　文件检索对话框

函数 uigetfile 可以创建文件检索对话框，其调用格式为：

```
[filename, pathname, filterindex] = uigetfile(filterspec, title)
[filename, pathname, filterindex] = uigetfile(filterspec, title, file)
[filename, pathname] = uigetfile(..., 'multiselect', selectmode)
```

参数说明：filename 是选择的文件名称。Pathname 是选择文件所在的路径。filterindex 是过滤扩展名的序号。filterspec 用于限定打开文件的扩展名。title 是对话框的名称。file 是显示在文件检索对话框文件名处的名称，是默认打开的文件名称。multiselect 是属性名，表示是否为多选方式。selectmode 是控制是否多选的参数，可选值为：on 表示允许多选，off（默认值）表示单选。

下面举例说明文件检索对话框的创建，执行程序后可以得到如图 18.39 所示的对话框。

```
 [filename, pathname, filterindex] = uigetfile({'*.m','MAT-files
(*.m)';'*.mat',...
   'Models (*.mat)'; '*.*',  'All Files (*.*)'}, 'Pick a
file','draw_all','MultiSelect', 'on')
```

18.4.11　写入文件而显示的检索框

函数 uiputfile 可以创建为写入文件而显示的检索对话框，其调用格式为：

```
[filename, pathname, filterindex] = uiputfile(filterspec, title);
[filename, pathname, filterindex] = uiputfile(filterspec, title, file);
```

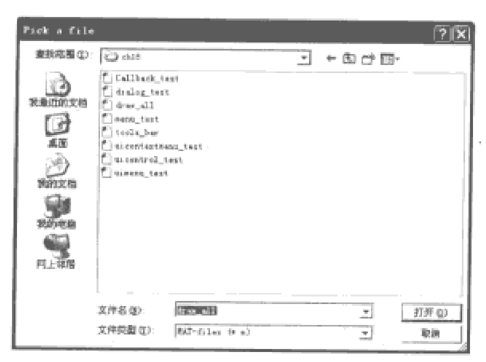


图 18.39 文件检索对话框

参数说明：filename 是写入的文件名称。Pathname 是写入文件所在的路径。filterindex 是过滤扩展名的序号。filterspec 用于限定写入文件的扩展名。title 是对话框的名称。file 是显示在检索对话框文件名处的名称，是默认写入的文件名称。

下面举例说明函数 uiputfile 的用法，执行下面语句可以得到如图 18.40 所示的图形。

```
[filename, pathname, filterindex] = uiputfile( {'*.mat','MAT-files (*.mat)';
'*.mdl','Models (*.mdl)';...
   '*.*', 'All Files (*.*)'}, 'Save as', 'Untitled.mat');
```

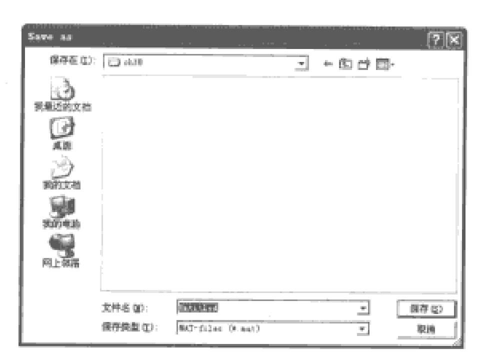


图 18.40 写入文件时显示的检索对话框

18.4.12 颜色设置对话框

函数 uisetcolor 可以创建颜色设置对话框，从而人机交互式设置对象的颜色，其调用格式为：

```
c = uisetcolor
c = uisetcolor([r, g, b])
c = uisetcolor(h)
c = uisetcolor(...,'dialogtitle')
```

参数说明：c 是选择的颜色向量，它是由 3 个元素组成的向量。r，g 和 b 是设定的默认颜色。h 是绘图对象的句柄。dialogtitle 是对话框名称。

下面举例说明函数 uisetcolor 的用法。执行下面的程序后，会弹出相应对话框。

```
ph=plot(rand(1,20));              % 绘制随机曲线
C = uisetcolor(ph,'Select a color')  % 创建颜色设置对话框
```

18.4.13 字体设置

函数 uisetfont 可以创建字体设置的对话框，从而实现人机交互设置对象字体信息，其调用格式为：

```
S = uisetfont(Fin, 'dialogTitle')
```

参数说明： S 是返回的已设置的字体信息，其为一个结构体数据。Fin 是含字体属性的图形对象，如函数 text，xlabel 和 legend 等得到的图形对象。dialogTitle 是对话框名称。

下面举例说明函数 uisetfont 的用法。执行如下程序后弹出的字体设置对话框如图 18.41 所示。

```
Fin=text(0.4,0.5,'ABCDEF');      % 标注字符
S = uisetfont(Fin, 'dialogTitle');   % 创建字体设置对话框
```

18.4.14 警告对话框

函数 warndlg 可以创建警告对话框，其调用格式为：

```
handle = warndlg('warnstring','dlgname','createmode')
```

参数说明： handle 是对话框的句柄。warnstring 是警告信息内容。dlgname 是对话框名称。createmode 用于指定对话框是否前置，其可选参数为 non-modal（非前置）和 modal（前置）。

下面举例说明函数 warndlg 的用法，执行下述语句可以得到如图 18.42 所示的对话框。

```
f = warndlg('This is an warning string.', 'My Warn Dialog', 'modal');
```

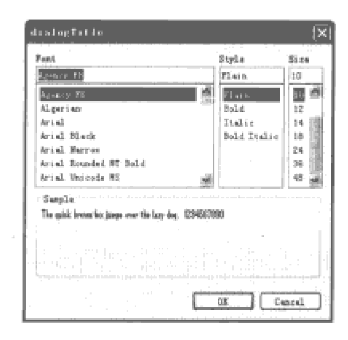


图 18.41 字体设置对话框



图 18.42 警告对话框



这里的警告对话框是前置的，即只有在警告对话框上面做出响应动作才能到当前 MATLAB 环境中进行其他操作。

18.4.15 计算进度条窗口

函数 waitbar 可以创建显示程序计算进度条窗口，其调用格式为：

```
H = waitbar(X,'message', property, value, property, value, ...)
waitbar(X,H)
waitbar(X,H,'message')
```

参数说明： H 是进度窗口的句柄。X 是一个 0~1 之间的数，表示程序已完成的部分（用红色

标记）。message 表示更新进度时显示的字符串。

下面举例说明函数 waitbar 的用法，如下程序执行后可以得到如图 18.43 所示的图形。

```
h = waitbar(0,'Please wait...');  % 创建进度条
for i=5:100,
   waitbar(i/200,h,char(i));   % 更新进度条的进度
end
```

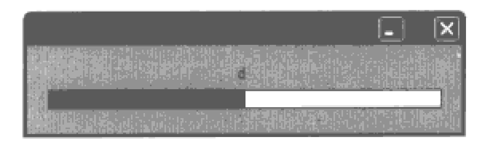


图 18.43　进度条



用户可以利用函数 close 来关闭进度条，即执行“close(h);”就可以自动关闭进度条。利用进度条可以很方便地了解大型程序计算的进度情况。

18.5　实例

本章前面介绍了利用 MATLAB 函数实现基本用户界面窗口控件的方法。此外 MATLAB 还提供了专门制作 GUI 的工具。本节首先介绍制作 GUI 的工具 GUIDE，然后举例说明创建图形界面窗口。用户在命令窗中选择菜单 File/New/GUI 后可以得到如图 18.44 所示的界面。

用户可以根据需要选择 GUIDE 模板。其中有 4 种可选样式 Blank GUI (Default)，GUI with Uicontrols，GUI with Axes and Menu 和 Modal Question Dialog，如图 18.45 和图 18.46 所示。从中可以看出，4 种类型的模板窗口左侧有两列控件板，利用鼠标可以把这些控件拖放到网格区域，还可以很方便地调整控件的位置和大小。控件的属性编辑只需在控件上单击鼠标右键即可弹出如图 18.47 所示的菜单，选择其中的 Property Inspector 命令就可以进行属性编辑，并弹出如图 18.48 所示的对话框。

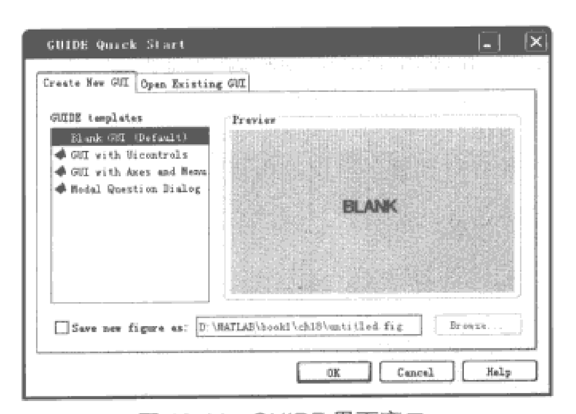


图 18.44　GUIDE 界面窗口

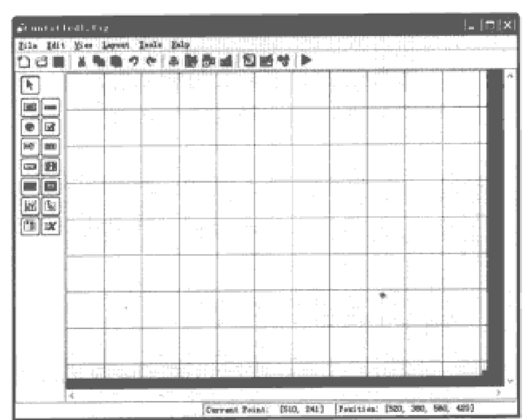
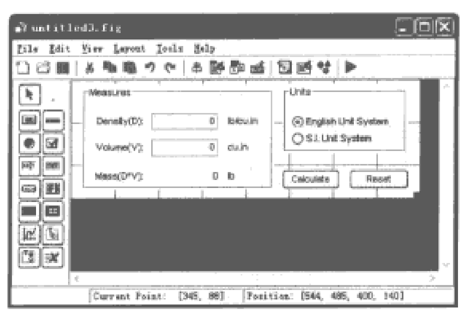
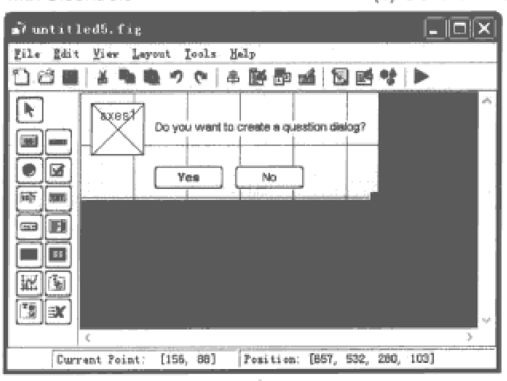


图 18.45　选择“Blank GUI (Default)”后的界面



(a) GUI with Uicontrols

(b) GUI with Axes and Menu



(c) Modal Question Dialog

图 18.46　带有控件的 3 种 GUIDE 设计模板

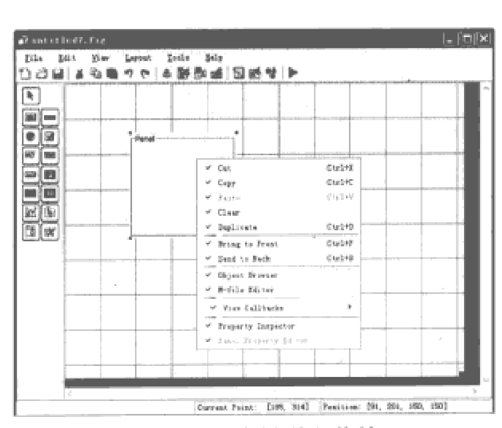


图 18.47　右键弹出菜单

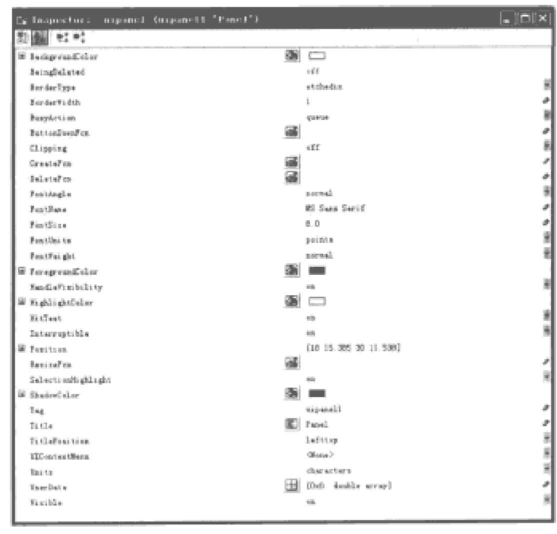


图 18.48　控件属性编辑对话框

如果保存 GUIDE 设计对象可以同时得到.fig 文件和一个同名的 M 文件。当需要对 GUI 进行编辑的时候，在图 18.44 中选择 Open Existing GUI 选项卡就可以编辑一个已经存在的 GUI。相应的 Callback 内容是在 M 文件中输入的。通过图 18.44 至图 18.48 所示的对话框就可以方便地建立一个图形界面窗口了，其效果和利用 uicontrol 函数制作的控件一样。不过利用 GUIDE 设计的 GUI 在版本兼容性方面可能会出现问题，程序的可移植性要比利用函数 uicontrol 制作的 GUI 差一些。

图 18.49 中界面上的控件是利用函数 uicontrol 生成的，滑动条可以用来改变贝塞尔函数的阶

数，同时在滑动条上还设置有鼠标提示信息，即“v=0.72”。(该程序保存在光盘的\Ch18 文件夹下，文件名为 Cures_GUI.m。) 用户可以模拟此例建立其他功能的图形界面窗口。

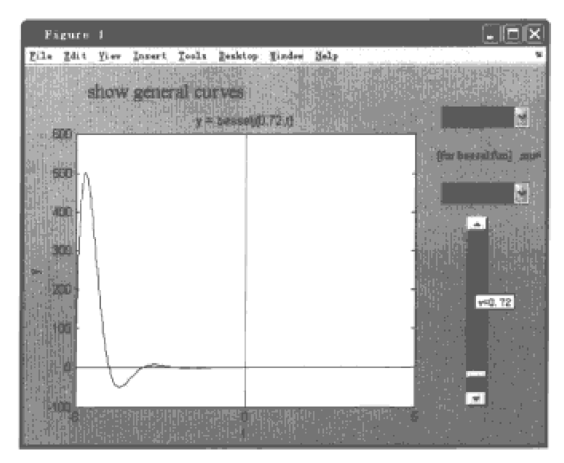


图 18.49　利用滑动条取值得到的结果

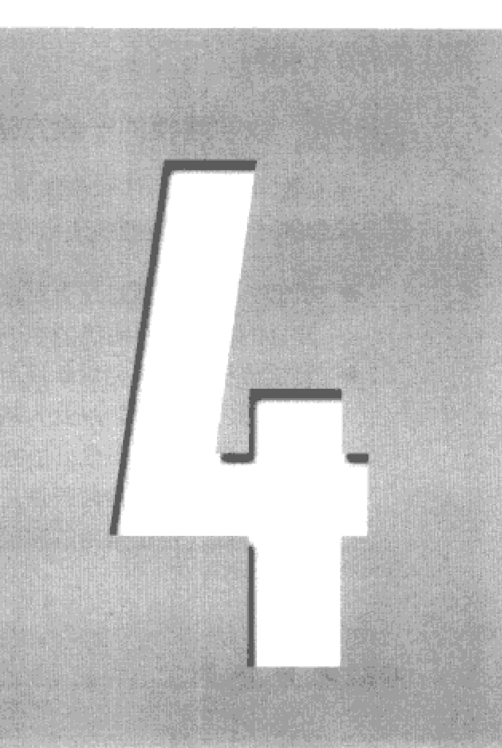
18.6　小结

本章主要介绍了图形界面窗口的设计方法。利用 MATLAB 可以设计出不同类型的控件、菜单、按钮以及对话框，利用 GUI 可以进行人机交互操作，对研究一些含有变化的参数、连续变化过程的问题有重要的帮助。本章首先介绍了窗口菜单、右键菜单和弹出菜单的制作方法。然后介绍了工具条、按钮和开关按钮的制作，如何利用函数 uicontrol 创建按钮、开关按钮、无线按钮、编辑框、列表框以及下拉菜单的方法。接下来介绍了不同类型对话框的设计，如错误对话框、帮助对话框、问题对话框以及进度条等的创建过程。最后结合 GUIDE 设计工具介绍了用户界面窗口的实例。

◆　　◆　　◆



第4部分　科学问题编程



第 19 章 MATLAB 建模基础

本章包括

- ◆ **抽象模型** 介绍数学模型的建立方法，并给出相应的实例分析。
- ◆ **离散采样方法** 结合实例介绍对于离散问题的处理方法。
- ◆ **算法结构设计** 举例介绍算法设计中的一些方法。
- ◆ **实例仿真** 举例说明利用 MATLAB 处理实际问题。
- ◆ **验证方法** 介绍验证程序正确性的方法。
- ◆ **算法优化** 举例介绍算法优化对程序执行效率的改善。

建立模型是解决科学与工程问题的一个重要中转环节。主要思路是利用数学化和过程性语言来描述或者翻译目标问题。根据这些数学化的描述和过程性语言，可以建立离散化的公式和解决问题的流程，从而利用科学计算软件来模拟和求解问题。本章介绍基本的建模知识，包括抽象模型、离散化问题、算法设计与优化等。

19.1 抽象模型

抽象模型是对科学和工程问题进行长时间思考而建立的一种抽象化的认识，可以借鉴数学建模的思想。数学模型是针于现实世界的特定对象或特定目的，根据特有的内在规律，做出一些必要的假设，运用适当的数学工具，得到的数学结构。简单地说，就是科学和工程问题中的某种特征本质的数学表达式（或是用数学语言对部分现实世界的描述），即用数学公式（如函数、图形、代数方程、微分方程、积分方程和差分方程等）来描述（表述、模拟）所研究的客观对象或系统在某一方面的存在规律。

19.1.1 数学建模的一般方法和步骤

建立数学模型的方法和步骤并没有一定的模式，但一个理想的模型应能反映系统的全部重要特征：模型的可靠性和模型的使用性。建模的一般方法有以下几种。

- ◆ 机理分析：根据对现实对象特性的认识，分析其因果关系，找出反映内部机理的规律，所建立的模型常有明确的物理或现实意义。
- ◆ 测试分析：将研究对象视为一个“黑箱”系统，内部机理无法直接寻求，通过测量系统的输入输出数据，并以此为基础运用统计分析方法，按照事先确定的准则在某一类模型中选出一个数据拟合得最好的模型。测试分析方法也叫做系统辩识。
- ◆ 将这两种方法结合起来使用，即用机理分析方法建立模型的结构，用系统测试方法确定模型的参数，也是常用的建模方法。

在实际过程中用哪一种方法建模主要是根据对研究对象的了解程度和建模目的来决定。机理分析法建模的具体步骤大致如下：

step 1 实际问题通过抽象、简化、假设确定变量和参数。
step 2 建立数学模型并数学、数值地求解、确定参数。
step 3 用实际问题的实测数据来检验该数学模型。
step 4 符合实际，交付使用，从而可产生经济、社会效益；不符合实际，重新建模。

19.1.2 数学模型的分类

按研究方法和对象的数学特征分初等模型、几何模型、优化模型、微分方程模型、图论模型、逻辑模型、稳定性模型、统计模型等。

数学建模需要丰富的数学知识，涉及到高等数学、离散数学、线性代数、概率统计以及复变函数等基本的数学知识。同时还要有深厚的专业基础知识、较强的逻辑思维能力以及语言表达能力等。

对于求解某一科学或者工程问题，如何着手往往是最困难的环节。建立模型的基本原则是认真分析目标问题的已知条件，找到与之相关的影响因素。这些影响因素可能是定量的（能够使用数量来描述），也可能是定性的。还应该进一步获得各个因素之间的一些简单函数关系。定量的因素可以分类为变量、参量及常量（如物理常数等）。参量对于一些特定问题可以是常量，而对于不同问题常量的值也可能不同。变量可以分为离散型和连续型，也可以分为确定型和随机型。

在实际问题中，结果往往受很多因素的影响，它们的影响程度可能不同。因此在分析各个因素与结果的关系后，首先应该考虑主要因素，忽略最次要的因素，在分析问题精力和计算能力允许的条件下考虑尽可能多的影响因素。而确定影响因素的主次关系往往比较困难，它要求对问题背景有深刻的理解和丰富的经验。有时一些因素被误认为是不重要的因素，可能对结果的影响比较大。

为了使建模顺利进行，需要进行一些合理的假设。其目的在于做出变量的取舍，即确定不同变量的重要程度。使目标问题得以简化便于利用数学语言描述，同时抓住目标问题的本质。假如我们把建模比做“建造大楼”，各个因素就好比是建筑材料，如砖块、钢筋、砂子等，而假设环节就是用水泥把各个因素组合在一起。在建模过程中，成功与否很大程度上依赖于所作假设的合理性，这个工作主要取决于建模研究者的理论基础和经验。假设可以分为两类：一类是为了简化科学或者工程问题的需要而做出的，另一类是为了沿用某一数学算法而做出的。具体的假设需要根据目标问题来确定。值得指出的是，检验假设是否合理需要依据假设是否满足所研究的实际问题，而不是为了解决问题的便利而歪曲原来的问题。

19.1.3 数学建模示例

下面举例说明建立抽象模型的过程。

例19-1：狗、鸡、白菜渡河问题：某人需要把随身所带的一只狗、一只鸡和一颗白菜运过河，而限制条件是现有的一条船除了人之外每次只能带一样东西过河。问题是这个人如何安排载渡顺序才能使得狗不吃鸡、鸡不吃白菜（注：有人在场时，狗不能吃鸡、鸡不能吃白菜）。

这是一道智力游戏问题，大家可以通过多次反复试验而得到正确的答案。现在的问题是如何利用数学语言给出一个逻辑推导过程。实际上大家在推算这个问题时所用的一些记录符号就是数学语言的雏形。

本题目描述的安全过河问题可以理解成一个多步决策的过程。在每一步决策中，即船在此岸和彼岸之间“往”或“返”之前，都需要对船和岸上的东西做出决策，在保证安全的前提下，在有限次往返过程中把所有物品运过河，即到达彼岸。数学上，可以把人、狗、鸡和白菜依次用一个由 4 个元素组成的向量表示（man，dog，chicken 和 cabbage），元素按顺序对应于人、狗、鸡和白菜。如果考虑对象（人、狗、鸡和白菜之一）在此岸，则对应元素为 1，若在彼岸则对应元素等于 0，比如（1，0，1，0）表示人和鸡在此岸，（0，0，1，1）表示鸡和白菜在彼岸。这里用 S 表示所有可能的状态，即：

```
(1, 1, 1, 1), (1, 1, 1, 0), (1, 1, 0, 0), (1, 1, 0, 1)
(1, 0, 0, 0), (1, 0, 0, 1), (1, 0, 1, 0), (1, 0, 1, 1)
(0, 1, 0, 0), (0, 1, 0, 1), (0, 1, 1, 0), (0, 1, 1, 1)
(0, 0, 0, 0), (0, 0, 0, 1), (0, 0, 1, 0), (0, 0, 1, 1)
```



下画线标记的向量表示允许的状态。其中（1，1，1，1）是初始状态，（0，0，0，0）是目标状态。这 16 个状态可以利用下面的 MATLAB 程序生成。

```
S='(0,0,0,0)';Ss=[];                              % 初始化字符串 S 和 Ss
for k=0:15;
   S(2:2:end)=strrep(num2str(bitget(k,1:4)),' ','');   % k 提取二进制数位并转化为字符串形式
   if k>2 & mod(k+1,4)==0;                        % 检查 k 是否为 3, 7, 11, 15
      Ss=[Ss,'、',S];                             % 更新 Ss 的数据
      disp(Ss(2:end));Ss=[];                      % 显示 Ss 的内容，同时清空 Ss 的内容
   else
      Ss=[Ss,'、',S];                             % 更新 Ss 的数据
   end
end
```

上述程序输出为：

```
(0,0,0,0), (1,0,0,0), (0,1,0,0), (1,1,0,0)
(0,0,1,0), (1,0,1,0), (0,1,1,0), (1,1,1,0)
(0,0,0,1), (1,0,0,1), (0,1,0,1), (1,1,0,1)
(0,0,1,1), (1,0,1,1), (0,1,1,1), (1,1,1,1)
```

如果把每次运载的情况也利用一个 4 元素向量表示，如（1，1，0，0）表示人和狗在船上而鸡和白菜不在船上，这样允许的运载状态 D 只有 4 种允许取值，即：

```
(1, 1, 0, 0), (1, 0, 1, 0), (1, 0, 0, 1), (1, 0, 0, 0)
```

这里规定 S 和 D 中的元素相加时按二进制法则进行来对应渡河过程。因此，单程过河就是一个允许状态向量 S 和一个允许运载向量 D 的相加运算。因此决策问题可以用数学语言描述为：求出决策 $d_k \in D$，使状态向量 $s_k \in S$ 按照运算规则从状态（1，1，1，1）经过有限次计算得到向量（0，0，0，0）。

这里考虑利用 MATLAB 程序来模拟上述过程。利用运载状态向量 D 作用于允许状态向量 S，保留允许的状态，删除一些不允许的状态。其中运载状态矩阵需要根据岸上的物品决定，比如岸上有狗和白菜，那么允许的运载状态向量有（1，1，0，0），（1，0，0，1），（1，0，0，0），即船上运载的内容可以是：“只有人”、“人和狗”、“人和白菜” 3 种可能。同时需要限制连续两次运载内

容不能一样，否则会进入死循环。下面给出实现本问题的模拟 MATLAB 程序。

```
s=[1,1,1,1];                    % 初始状态
Ss=s;                          % 记录状态变化的数组
D=[1,0,0,0;1,1,0,0;1,0,1,0;1,0,0,1];  % 运载状态组成的矩阵
Ds=[];                         % 记录运载状态变化的数组
v=[8,4,2,1];                    % 二进制转化为十进制数的基
Vi=[3,6,7,8,9,12];                % 不允许状态向量对应的十进制数
k=1;                           % 标记往返的参数
while sum(s)>0.5;
    if mod(k,2)==1;            % 判断往返情况，1 表示往，0 表示返
        x=find(s==0);          % 找出在彼岸的物品
    else
        x=find(s==1);          % 找出在此岸的物品
    end
    Dt=D;Dt(x,:)=[];            % 计算出可能的状态矩阵
    if ~isempty(Ds);
        Dt=setdiff(Dt,Ds(end,:),'rows'); % 删去上一步用过的运载状态向量，避免死循环
    end
    St=repmat(s,size(Dt,1),1);     % 复制表示状态的向量使矩阵 St 和矩阵 Dt 的行数和列数相等
    SD=mod(Dt+St,2);             % 通过加法规则计算运载后的结果
    Sv=sum(SD.*repmat(v,size(Dt,1),1),2);  % 计算运载后状态向量对应的十进制数
    V=setdiff(Sv,Vi','rows');        % 从运载后的状态中删去不允许的状态
    x=find(Sv==min(V));            % 找出一个允许的运载状态
    s=SD(x,:);                   % 得到运载后的状态向量
    Ds=[Ds;Dt(x,:)];               % 把当前运载状态向量加入到矩阵 Ds 中
    Ss=[Ss;s];                   % 把当前状态向量加入到矩阵 Ss 中
    k=k+1;                      % 更新标记往返的参数
end
Process=[Ss(1:7,:),Ds]           % 输出状态向量及对应的运载状态向量，其中两部分合为一个矩阵显
示 Process
s                              % 输出最后结果
```

运行上述程序，输出结果为：

```
Process =
     1    1    1    1     1    0    1    0
     0    1    0    1     1    0    0    0
     1    1    0    1     1    1    0    0
     0    0    0    1     1    0    1    0
     1    0    1    1     1    0    0    1
     0    0    1    0     1    0    0    0
     1    0    1    0     1    0    1    0
s =  0    0    0    0
```

结果表明经过 7 次运载，可以把所有物品运过岸。



在显示的过程矩阵 Process 中，左侧划波浪线的部分是状态向量，右侧划下画线的部分是运载状态向量。用户可以根据前面说明解读上面的数据。

前面通过不同类型的例子说明了如何建立模型、程序模拟，用户可以从中学习解决实际问题的方法：分析问题背景、数学语言描述、程序模拟。在前面章节介绍了 MATLAB 数值计算与数据可视化的方法，利用这个功能强大的工具可以解决很多科学与工程问题。在求解问题过程中，对于获

得的结果还需要从专业知识和实际结果出发来验证结果的可靠性。

19.2 离散采样方法

对于离散的数据而言，需要根据数据的变化规律得出相应的经验模型。在这样的经验模型中，变量之间的关系通过数据变化特点而得到数学公式，所得表达式比前面介绍的抽象模型要简单同时有一定的准确性。数据与理论曲线之间的误差可以看做是一些未知因素的影响。所得函数公式不是来自于事先假设，也不是基于物理上的规律或者原理，而是通过建模者认为已知数据与某类数学公式之间具有很大的相似性而选取的。这样的模型经常是复杂模型的子模型。得到经验模型的步骤如下：

step 1 把已知数据绘制在坐标轴上，通过离散数据的走势来推断其所属数学函数类型。其中选择数学公式的优劣直接影响到经验模型的精确程度。

step 2 确定数学公式中的参数。

step 3 根据数学计算各点的函数值，并与已知数据相比较，得到二者的误差量。如果不符合精度要求，需要对数学函数的表达式重新认定或者重新计算公式中的参数。其中参数的确定可以利用最小二乘原理或者前面介绍的回归分析法。

例 19–2：分析身高与体重。

表 19.1 中的数据给出在某地测量 15 个不同年龄的人的身高与体重的数据。试分析其中的规律。

表 19.1 不同人的身高与体重表格

身高(cm)	体重(kg)	身高(cm)	体重(kg)	身高(cm)	体重(kg)
75	11	126	28	163	53
86	13	135	36	167	55
95	15	151	42	171	61
108	17	155	47	178	67
112	22	160	51	185	76

step 1 根据表 19.1 绘制数据相应的曲线，其中 X 轴为身高 H，Y 轴对应着体重 W。

step 2 具体的绘图程序如下：

```
H=[75,86,95,108,112,126,135,151,155,160,163,167,171,178,185]; % 输入身高数据
W=[11,13,15,17,22,28,36,42,47,51,53,55,61,67,76];              % 输入体重数据
s1=subplot(121);plot(H,W,'k*');axis square;                   % 绘图并设置当前坐标轴为
方形
xlabel('{\itH}','Fontname','Times New Roman');                % X 轴标注
ylabel('{\itW}','Fontname','Times New Roman');                % Y 轴标注
subplot(122);plot(H,log(W),'k*');axis square;                  % 绘图并设置当前坐标轴
为方形
xlabel('{\itH}','Fontname','Times New Roman');                 % X 轴标注
ylabel('ln {\itW}','Fontname','Times New Roman');              % Y 轴标注
```

step 3 输出图形如图 19.1 所示。

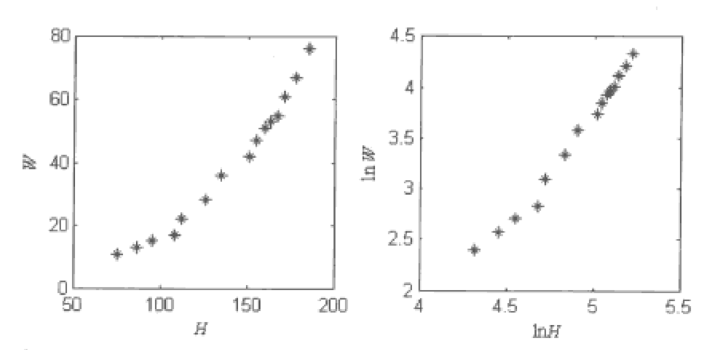


图 19.1　身高与体重的对应关系数据

step 4 从图 19.1 可以看出数据点接近于指数曲线，而通过绘制 lnH 对 lnW 的曲线可以进一步看出与指数函数吻合得很好。利用最小二乘法拟合数据有：

```
P = polyfit(log(H),log(W),1)
```

输出为：

```
P =    2.2385   -7.4374
```

step 5 从而可以得出身高和体重的关系表达式：$\ln W = 2.2385\ln H - 7.4374$。两边取 e 指数有：$W = 5.8881H^{2.2385}$。利用拟合的公式计算，得到的曲线如图 19.2 所示。

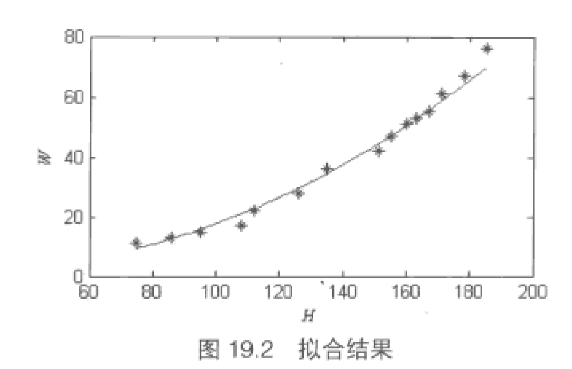


图 19.2　拟合结果

计算的 MATLAB 程序代码如下：

```
P = polyfit(log(H),log(W),1); % 进行多项式拟合
h=linspace(75,185);% 生成等间隔采样数据
plot(H,W,'*');hold on;% 绘制已知数据曲线
plot(h,5.8881e-004*h.^2.2385,'r');% 绘制拟合曲线
xlabel('{\itH}','Fontname','Times New Roman'); % X 轴标注
ylabel('{\itW}','Fontname','Times New Roman'); % Y 轴标注
```

例 19-3：分析潮水高度。

表 19.2 给出了某地连续 24 小时记录的潮水高度数据，试建立经验模型的数学公式。

表 19.2　潮水高度数据表

时间	高度(m)	时间	高度(m)	时间	高度(m)	时间	高度(m)
00:00	2.7	06:00	-2.9	12:00	3.2	18:00	-3.3

（续表）

时间	高度(m)	时间	高度(m)	时间	高度(m)	时间	高度(m)
01:00	1.8	07:00	-2.1	13:00	2.0	19:00	-2.8
02:00	0.3	08:00	-0.3	14:00	0.6	20:00	-1.2
03:00	-1.1	09:00	1.6	15:00	-0.9	21:00	0.8
04:00	-2.3	10:00	3.2	16:00	-2.2	22:00	2.6
05:00	-3	11:00	3.7	17:00	-3	23:00	3.5

step 1 根据数据绘制二维图形，相应程序如下：

```
T=0:23;                % 时间数据
H=[2.7,1.8,0.3,-1.1,-2.3,-3,-2.9,-2.1,-0.3,1.6,3.2,3.7,3.2,2.0,0.6,-0.9,-2.2,-
3,-3.3,-2.8,-1.2,0.8,2.6,3.5];
figure;plot(T,H,'k*')      % 绘图
xlim([min(T),max(T)]);  % 设置 X 轴范围
```

step 2 输出图形如图 19.3 所示。

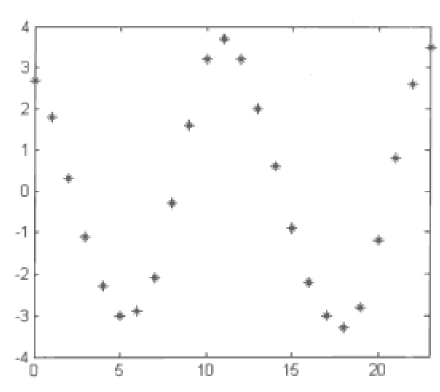


图 19.3 潮水高度随时间变化的数据图

step 3 从图 19.3 中数据分布的特点可以看出数据分布接近于函数关系 $h = a\sin(bt + c)$。确定了函数的参数 a， b 和 c 就得到了经验公式。

```
xdata=0:23;     % 输入待拟合的数据
ydata=H;      % 输入高度数据
options=optimset('TolFun',1e-6);  % 设置优化选项
x = lsqcurvefit(@(x,xdata)
x(1)*sin(xdata*x(2)+x(3)),[4.1,pi*2/18,1.1],xdata,ydata,[],[],options)
t=linspace(0,23);          % 生成等间隔采样数据
figure;plot(T,H,'k*');     % 绘图
hold on;
plot(t,x(1)*sin(x(2)*t+x(3)),'r')   % 绘制拟合曲线
```

step 4 输出结果如下：

```
x =    -3.3657    0.5117   -1.0854
```

step 5 因此经验公式可以写为：$h = -3.3657\sin(0.5117t - 1.0854)$。根据经验公式绘制的曲线如图 19.4 所示。

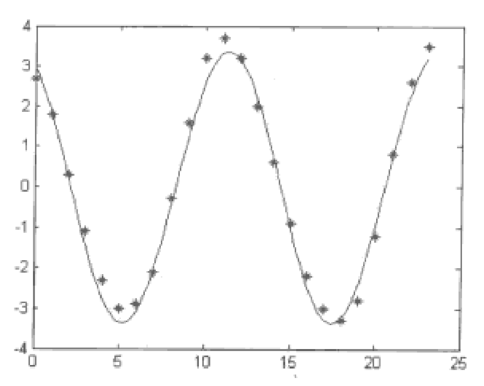


图 19.4 根据潮水高度数据拟合的曲线

19.3 算法结构设计

在前面两节介绍的模型建立方法的基础上，可以考虑建立程序来模拟一些过程或者计算一些结果。在建立程序的过程中程序结构设计非常重要，需要按照一定的顺序来计算相关的变量，下面举例来说明程序设计。

例 19–4： 分析狗运动的轨迹。在一个广场上，一个人正在以匀速度 v_m 沿一直线跑步。此时一只狗从某一位置开始以速率 v_d 时刻瞄准人开始追赶，如图 19.5 所示。

问题的关键在于狗的运动方向在不断地变化，这里考虑建立一个动画来模拟这个过程。本问题中的参数有 4 个，即初始时人和狗的位置 p_m 和 p_d，以及它们的速度。在设计程序时可以考虑不同的情况进行分析。下面给出求解这个问题的步骤：

step 1 给定问题的参数，同时考虑所有可能的情况。
step 2 以时间 dt 为单位计算各个离散时刻人和狗的位置。
step 3 计算当前狗的速度，并给出下一时刻狗的位置。
step 4 如此循环下去就得到一系列位置信息，从而得到狗运动的轨迹。

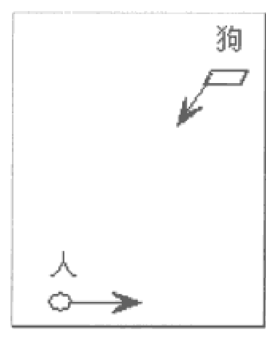


图 19.5 狗追人模型

根据上面的分析，假定人的运动速度是 3 米/秒，狗的速率是 3.5 米/秒，给出相应的模拟程序。

```
zm=i;                      % 人的位置
zd=8+5i;                    % 狗的位置
vm=3;                      % 人的速度
```

```
vd=3.5;                   % 狗的速度
dt=0.02;                  % 时间间隔
A=0;                                % 设置人运动的方向
pm=plot(zm,'ko');                       % 绘制人的位置
hold on;
pd=plot(zd,'k');                        % 绘制狗运动的轨迹
axis([0,8,0,5]);                         % 设置坐标轴范围
while abs(zm(end)-zd(end))>0.01;           % 循环计算狗的运动轨迹
    At=angle(zm-zd(end));               % 计算狗当前运动的方向
    zd=[zd,zd(end)+exp(i*At)*vd*dt];    % 计算狗在下一时刻的位置
    zm=zm+exp(i*A)*vm*dt;             % 计算下一时刻人的位置
    set(pm,'xdata',real(zm),'ydata',imag(zm));  % 更新人的位置
    set(pd,'xdata',real(zd),'ydata',imag(zd));    % 更新狗的运动轨迹
    pause(0.2);                           % 暂停 0.2 秒显示动画效果
end
```

当程序停止时，输出图形如图 19.6 所示。

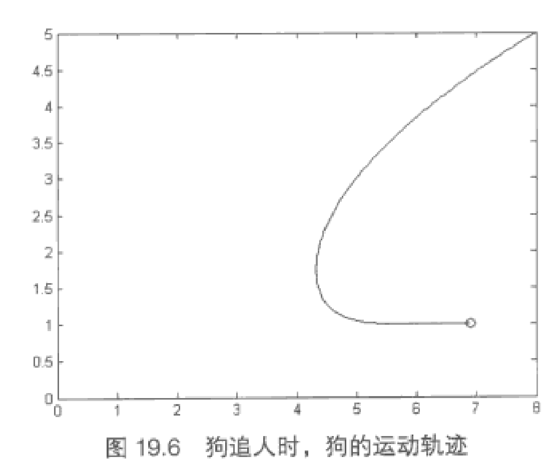


图 19.6 狗追人时，狗的运动轨迹

这里可以进一步考虑狗取不同速度时的结果，把上述程序中关于人和狗速度赋值的语句注释掉，得到关于速度的函数文件，对应光盘中\Ch19 文件夹下的 dog_trace.m 文件。该函数的调用格式为：

```
dog_trace(vm,vd);
```

参数说明： vm 是人的速度，vd 是狗的速度。

下面调用 dog_trace 来计算不同狗速度的轨迹结果。

利用下面的程序计算不同狗速度对应的轨迹曲线。

```
s(1)=subplot(131);dog_trace(3,3.2);% 绘制 vd=3.1m/s 时的轨迹
s(2)=subplot(132);dog_trace(3,3.7);% 绘制 vd=3.7m/s 时的轨迹
s(3)=subplot(133);dog_trace(3,4.2);% 绘制 vd=4.2m/s 时的轨迹
axis(s,'square');                    % 设置坐标轴为方形
```

上述程序输出结果如图 19.7 所示。用户还可以计算当人运动速度方向选取不同数值时的结果。下面考虑当人运动的轨迹是圆的时候，狗运动时留下的轨迹，相应程序如下。

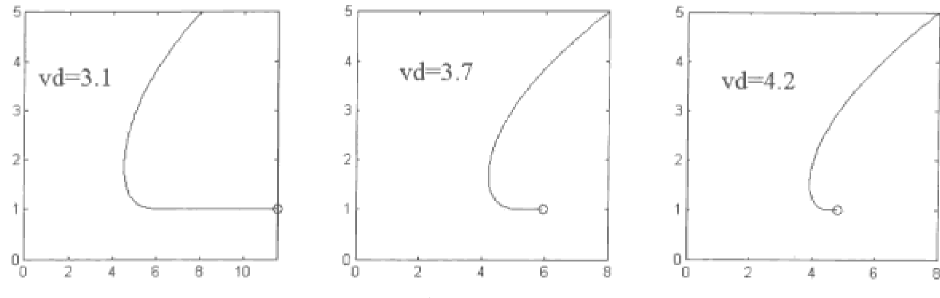


图 19.7　狗速度不同时狗运动的轨迹图

```
vm=3;     % 人的速度
vd=3.1;    % 狗的速度
zm=6i;     % 人的位置
zd=0;       % 狗的位置
dt=0.02;    % 时间间隔
A=0;         % 设置开始时人运动的方向
R=abs(zm);  % 圆轨迹的半径
dA=vm*dt/R; % 角速度
pm=plot(zm,'ko');  % 绘制人的位置
hold on;
pd=plot(zd,'k');      % 绘制狗运动的轨迹
axis([-R,R,-R,R],'square');          % 设置坐标轴范围
while abs(zm(end)-zd(end))>0.01;  % 循环计算狗的运动轨迹，狗追上人，它们都停止运动
   At=angle(zm-zd(end));          % 计算狗当前运动的方向
   zd=[zd,zd(end)+exp(i*At)*vd*dt];  % 计算狗在下一时刻的位置
   A=A+dA;                        % 更新角度值
   zm=R*exp(i*A);  % 计算下一时刻人的位置
   set(pm,'xdata',real(zm),'ydata',imag(zm));  % 更新人的位置
   set(pd,'xdata',real(zd),'ydata',imag(zd));    % 更新狗的运动轨迹
   pause(0.2);                            % 暂停 0.2 秒显示动画效果
end
```

运行上述程序，最终输出结果如图 19.8 所示。

前面介绍了人沿着直线或者圆运动的情况，这两种情况可以用匀线速度和角速度来更新人的位置。如果人运动的轨迹不是这样的轨迹，比如方形，该如何分析呢？需要明确的是如何从人运动的速度来计算人的位置。这里通过计算人走过路途的长度来确定人的位置，假定开始时人处于[0, L]，其中 L 是方向的半边长，如图 19.9 所示。

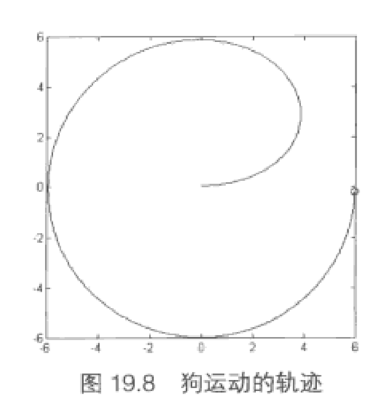


图 19.8　狗运动的轨迹

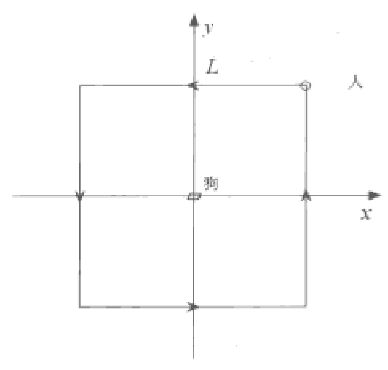


图 19.9　人沿着方形跑道运动

这里假定人开始时位于(L,L)点处。设人跑过的长度为S，下面考虑根据S确定人处于何处的计算方法。人在方形轨道上是一个周期性运动，S可以对其周期$8L$进行模（mod）计算，即$S_t=\mathrm{mod}(S,8L)$，其中$S_t\in[0,8L)$。

这样S_t就是一个周期内的长度了。根据S_t分属区间的不同可以得到如下一个分段函数：

$$x(S_t)=\begin{cases}L-re, & 0\le S_t<2L\\ -L, & 2L\le S_t<4L\\ re-L, & 4L\le S_t<6L\\ L, & 6L\le S_t<8L\end{cases} \tag{19-1}$$

$$y(S_t)=\begin{cases}L, & 0\le S_t<2L\\ L-re, & 2L\le S_t<4L\\ -L, & 4L\le S_t<6L\\ re-L, & 6L\le S_t<8L\end{cases} \tag{19-2}$$

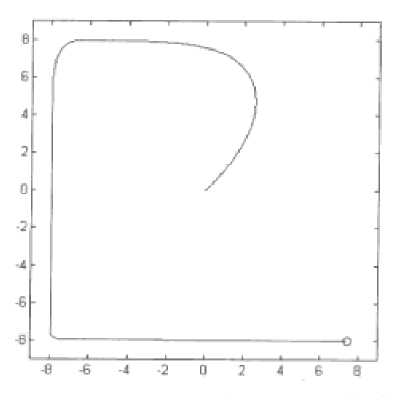
即同一条边上可以根据S_t计算得到确定的坐标值，上面的表达式是分段函数。其中re表示S_t和$2L$相除的余数，用 MATLAB 命令表述就是 re = rem(St,2L)。根据上面的分析，给出这种情况下的实现程序，如下：

```
vm=3;                     % 人的速度
vd=3.05;                  % 狗的速度
L=8;
zm=L+L*i;                 % 人的位置
S=0;                      % 初始时人走过的距离
zd=0;                     % 狗的位置
dt=0.02;                  % 时间间隔
pm=plot(zm,'ko');         % 绘制人的位置
hold on;
pd=plot(zd,'k');          % 绘制狗运动的轨迹
P=[L,-L,-L,L];
axis([-(L+1),(L+1),-(L+1),(L+1)],'square');  % 设置坐标轴范围
while abs(zm-zd(end))>0.01;  % 循环计算狗的运动轨迹，狗追上人，它们都停止运动
    At=angle(zm-zd(end));  % 计算狗当前运动的方向
    zd=[zd,zd(end)+exp(i*At)*vd*dt];   % 计算狗在下一时刻的位置
    S=S+dt*vm;            % 更新角度值
    St=mod(S,8*L);        % 计算变量 St 的数值
    re=rem(St,L*2);       % 计算 St 和 2L 相除的余数
    n=fix(St/2/L);        % 计算 St 和 2L 相除的商数
    switch n;
        case 0;
            x=L-re;       % 人处于上面一条边上的 X 轴坐标值
            y=L;          % 人处于上面一条边上的 Y 轴坐标值
        case 1;
            x=-L;         % 人处于左面一条边上的 X 轴坐标值
            y=L-re;       % 人处于左面一条边上的 Y 轴坐标值
        case 2;
            x=re-L;       % 人处于下面一条边上的 X 轴坐标值
            y=-L;         % 人处于下面一条边上的 Y 轴坐标值
        case 3;
            x=L;          %人处于右面一条边上的 X 轴坐标值
            y=re-L;       % 人处于右面一条边上的 Y 轴坐标值
    end
    zm=x+i*y;                          % 得到下一时刻人的位置
```

```
    set(pm,'xdata',real(zm),'ydata',imag(zm));  % 更新人的位置
    set(pd,'xdata',real(zd),'ydata',imag(zd));    % 更新狗的运动轨迹
    pause(0.2);                                  % 暂停0.2秒显示动画效果
end
```

输出结果如图19.10左图所示。

当把前面程序中的“zm=L+L*i;”和“S=0”换为“zm=L*i;”和“S=L”时，就可以模拟人初始时位于(0,L)这一点开始的运动情况，此时所得狗的运动轨迹如图19.10右图所示。



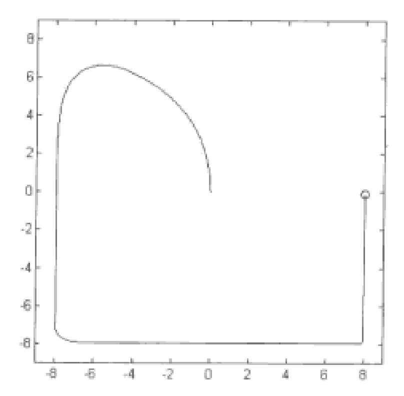


图19.10 人沿方形轨道运动时狗的运动轨迹

例19-5： 有的时候一些专门的软件可以每隔一段时间产生一个数据文件，但是这些软件可能不提供相应的数据处理程序。为此需要编写一个专门的数据小程序来实时地处理数据，这样就可以便于进行下一步的数据分析工作。

这个问题明确需要一个循环结构，比较好的就是while语句实现的不定次数的循环。可以利用下面的格式：

```
while 1
    % 这里是程序计算部分
End
```

这样程序可以一直执行下去，直至用户按“Ctrl+C”组合键强行中止程序执行。在循环内部需要实时地检测数据文件是否存在。检测数据文件是否存在可以用如下方法。

```
A=dir('*.ext');
names=A.name;
```

参数说明：ext是数据文件的扩展名，此时其他文件将不被检索。A是一个结构体，表示返回的文件信息。其中文件名信息可以利用names变量进行存储。

用户可以根据names变量的取值情况来确定数据文件的存在性。同时希望处理后的数据文件删除，如果数据文件用户不想保留，可以直接使用函数delete进行删除；如果需要保留数据文件，可以使用函数copyfile把数据文件保存到另外一个文件夹中，然后在当前文件夹把已经处理过的数据文件删除。

数据读入可以考虑MATLAB提供的函数load，textread，importdata，xlsread以及fread等。如果产生数据的软件输出的数据类型可以选择，用户可以设定该软件和MATLAB都支持的文件类型来实现二者的数据交流。根据以上分析，可以建立如图19.11所示的实时数据处理结构。

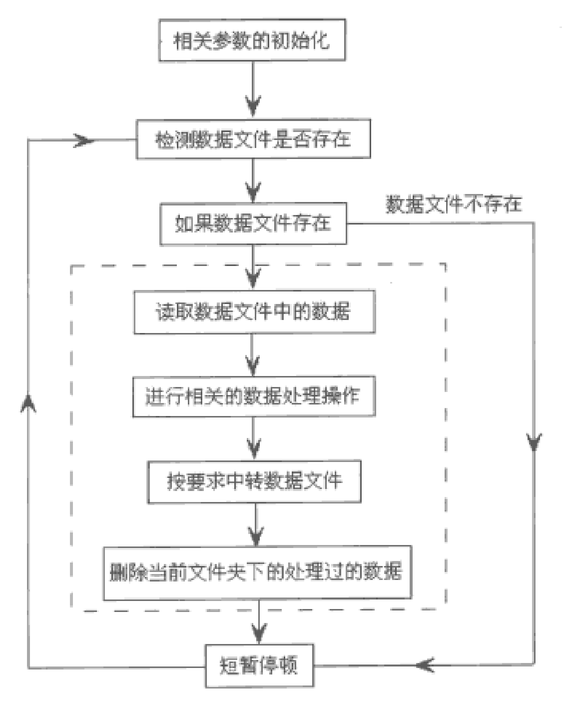


图 19.11　实时处理数据的程序结构



短暂停顿可以使 MATLAB 的 CPU 占用延缓一下，停顿的时间要根据数据处理的时间和产生软件生成相邻两个数据的时间间隔而定，它们达到匹配是最好的。这里需要对数据存在与否进行判断，如果数据不存在，直接进入停顿环节即可，其实现可以通过 if 语句。

19.4 实例仿真

根据前面介绍的方法，本节举例说明利用 MATLAB 解决一些数据问题的过程。

例 19-6：椭圆与直线相切的问题。椭圆的参数方程可以描述为：

$$\begin{cases} x = a\cos t \\ y = b\sin t \end{cases}，其中 0<b<a，t\in[0,2\pi) \tag{19-3}$$

现在有一水平直线方程 $y=-h$，其中 $b<h<a$，曲线位置关系如图 19.12 所示。求椭圆旋转多少角度时与直线相切。

首先分析这个问题的解析解，假设旋转角度为 β 时椭圆与直线相切，坐标旋转矩阵 T 为：

$$T=\begin{bmatrix} \cos\beta & \sin\beta \\ -\sin\beta & \cos\beta \end{bmatrix} \tag{19-4}$$

利用矩阵 T 可以计算出旋转后的椭圆参数方程为：

$$\begin{cases} x = a\cos t\cos\beta - b\sin t\sin\beta \\ y = a\cos t\sin\beta + b\sin t\cos\beta \end{cases} \tag{19-5}$$

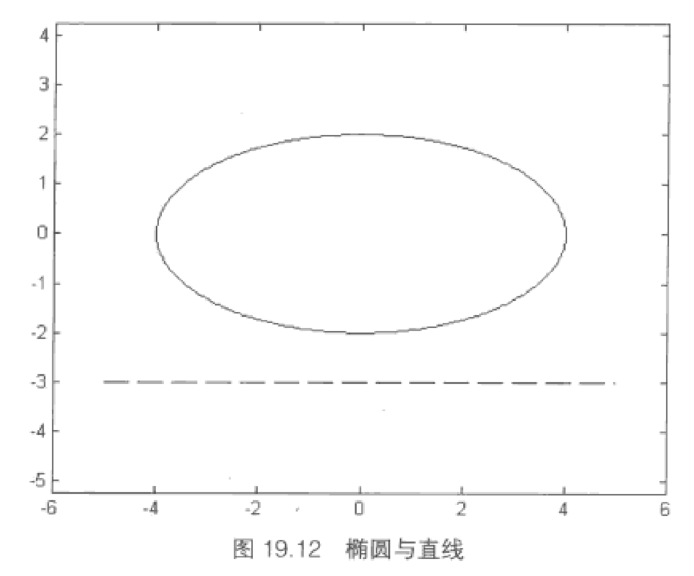


图 19.12 椭圆与直线

可以知道 y 的最小值是：

$$y_{\min} = -\sqrt{a^2\sin^2\beta + b^2\cos^2\beta}$$

当相切时，y 的最小值应该等于 $-h$，这样可以得到：

$$h^2 = a^2\sin^2\beta + b^2\cos^2\beta \tag{19-6}$$

因此可以得到：

$$\sin^2\beta = \frac{h^2-b^2}{a^2-b^2} \tag{19-7}$$

若限定 $\beta\in[0,\pi/2)$，则有：

$$\beta = \arcsin\left(\sqrt{\frac{h^2-b^2}{a^2-b^2}}\right) \tag{19-8}$$

设 $b=2$，$h=3$ 和 $a=4$，则 $\beta=\arcsin\left(\sqrt{5/12}\right)=0.7017$。

下面考虑利用 MATLAB 数值计算这个计算过程，采用的算法是二分法。首先把旋转前椭圆曲线上各点的坐标离散化，再考虑旋转角度的上下限，分析可知旋转角度介于 0 到 $\pi/2$ 这个范围，其中 0 对应着曲线和直线相离（曲线上最小值大于-h），而 $\pi/2$ 对应着椭圆和水平直线相交（曲线上最小值小于-h）。进行角度二分并分析椭圆与直线的位置关系，不断二分下去就可以得到旋转的角度。相应的实现程序如下：

```
t=linspace(pi,pi*2,401);% 离散化参数 t
a=4; % 对半长轴赋值
b=2; % 对半短轴赋值
h=3; % 对参数 h 赋值
P=[a*cos(t);b*sin(t)];% 把坐标点写为 2xN 的矩阵
b1=0; % 旋转角度的下限
b2=pi/2; % 旋转角度的上限
while abs(b1-b2)>1e-5;
    bm=[b1+b2]/2;% 取二等分点
    T=[cos(bm),sin(bm);-sin(bm),cos(bm)]; % 计算旋转矩阵
    Pm=T*P; % 计算旋转后坐标点对应的 2xN 的矩阵
```

```
    ym=min(Pm(2,:)); % 获得离散情况下的最小值
    if ym>-h; % 判断椭圆与直线是否相离
        b1=bm; % 用中值 bm 更新 b1 的值
    else
        b2=bm; % 对于相交情况，用中值 bm 更新 b2 的值
    end
end
bm=[b1+b2]/2 % 输出中值结果作为旋转角度的近似值
```



这里对参数 t 只取[π,2π]，因为椭圆与直线的位置关系可以通过下半部分的椭圆来考虑，甚至利用右下 1/4 部分也可，此时 t 的取值范围是[3π/2,2π]，如此可以减小离散化 t 的步长，从而增加计算精度。

上述程序输出结果为：

```
bm =
    0.7017
```

可见利用二分法也可以得到一个比较精确的结果，其结果与解析结果一致。特别是对于由复杂函数生成的曲线，利用离散方法计算就显示出优越性。

例 19-7：编写计算圆内任意两条弦相交的概率。如图 19.13 所示，两条弦利用虚线表示。弦 AB 的端点可以用角度 a_1 和 β_1 来表示。弦 CD 的端点可以用角度 a_2 和 β_2 来表示。

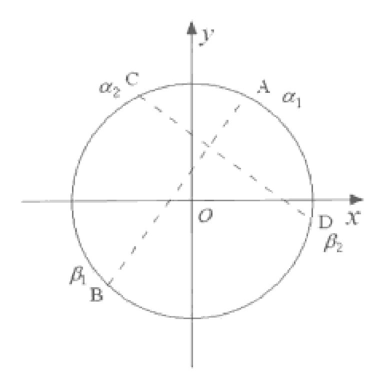


图 19.13　圆内的两条弦

假定这里考虑的圆的半径是 1，角度 a_1，β_1，a_2 和 β_2 表示单位圆上复数的复角，考虑下面的关系式：

$$F_1=(\alpha_2-a_1)(\alpha_2-\beta_1)\text{，}\quad F_2=(\beta_2-a_1)(\beta_2-\beta_1) \tag{19-9}$$

根据图 19.13 中所示的位置关系，F_1 小于 0 表示 C 点在弦 AB 的上侧，F_2 大于 0 表示 D 点在弦 AB 的下侧。弦 CD 与弦 AB 相交的充要条件是 C 和 D 两点在弦 AB 的两侧，即 $F_1F_2\le 0$，其中取等号对应着两条弦有公共定点。下面利用随机数来多次统计不同情况下的弦相交情况，程序如下：

```
N=1000000;                  % 实验总次数
intersect=0;                % 统计相交次数
for k=1:N;
    alpha1=rand*pi*2;       % 随机产生一个复角
    alpha2=rand*pi*2;       % 随机产生一个复角
```

```
    beta1=rand*pi*2;        % 随机产生一个复角
    beta2=rand*pi*2;        % 随机产生一个复角
    F1=[alpha2-alpha1]*[alpha2-beta1]; % 计算位置关系式
    F2=[beta2-alpha1]*[beta2-beta1];  % 计算位置关系式
    intersect=intersect+(F1*F2<=0);   % 如果相交，则相交次数累加1
end
format long       % 输出更多数位
p=intersect/N     % 输出弦相交的几率
```

输出结果为：

```
p =   0.333442000000000
```

19.5 验证方法

在进行数值模拟的过程中，结果的准确性和可靠性需要通过科学的方法来验证。通过验证可以知道程序是否编写正确以及编程前设计的算法是否合理。本节给出几种常用的验证方法。

◆ 理论结果验证。

根据已知的解析结果，在模拟程序得出结果之后二者进行比对。如果相等说明程序可能是正确的，因为有时用户编写的程序可能在某些特殊情况下是正确的。在实际中，一个成熟的程序需要在多个不同情况下验证通过才可以放心使用。

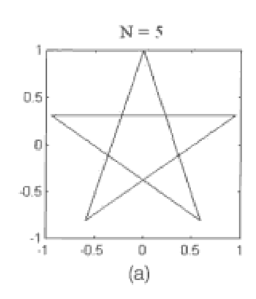
◆ 变化程序中利用关键参数来查看结果的正确性。

利用下面一段程序可以绘制五角星：

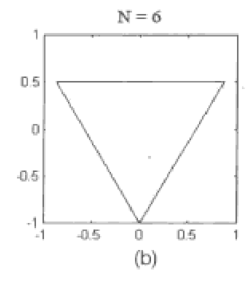
```
N=5;                          % 输出多角星的顶点数
A=pi*2*[0:N-1]/N+pi/2;        % 五等分点的角度值
n=0:2:2*N;                    % 生成绘制五角星的脚标索引
nt=mod(n-1,N)+1;              % 将大于N的脚标映射到不大于N的正整数
plot(exp(i*A(nt)));           % 绘图
```

输出图形如图19.14(a)所示。

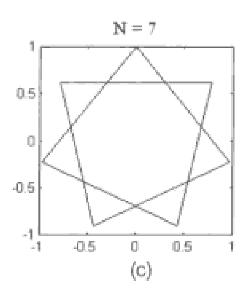
当把上面程序中的参数N改为6或7时，可以得到如图19.14(b)和(c)所示的图形。可见上述程序绘制六角星和七角星时得到的结果不正确。通过简单的更换关键参数N就可以知道这个程序的普适性，这说明这段程序只适用于绘制五角星，而其他多角形的绘制需要做进一步考虑，关于本问题的正确程序读者可以参阅前面小节的内容。



(a)



(b)



(c)

图19.14 多角星的绘制

◆ 依据问题要求验证结果的正确性。

```
x=solve('sin(x)-x/12')
```

比如求解方程 $\sin x - x/12 = 0$，可以使用下面的语句来验证根的正确性，即计算 sin(x)-x/12 的结果，如果结果等于 0，表示结果是方程的根。此外方程根的求解函数 solve 所得根不完整，需要通过作图法和函数 fzero 来求出遗漏的根。

◆ 利用经验或者专业知识来判断结果。

这一点需要用户从整体的层面考虑问题，对解可能出现的范围有一定的了解。下面是一个双摆动画的程序，用户可以通过对双摆运动的了解验证动画过程实现程序的正确性。

```
m1=1; % 上摆质量
m2=1; % 下摆质量
L1=1; % 上摆长度
L2=1; % 下摆长度
g=9.8;% 重力加速度
Da=inline(['[x(3);x(4);','inv([(m1+m2)*L1,m2*L2*cos(x(1)-x(2));',...

'm1*L1*cos(x(1)-x(2)),m1*L2])*','[m2*L2*x(4)^2*sin(x(2)-x(1))-(m1+m2)*g*sin(x(
1));',...
    'm2*L1*x(3)^2*sin(x(1)-x(2))-m2*g*sin(x(2))]]'],'t','x',...
    'flag','m1','m2','L1','L2','g'); % 定义双摆运动的微分方程
set(gcf,'DoubleBuffer','on'); %设置坐标轴渲染效果
[t,x]=ode45(Da,linspace(0,10,400-1),[0.8,0.8,0,0],[],m1,m2,L1,L2,g); % 求解双摆运
动微分方程
axis([-(L1+L2),(L1+L2),-(L1+L2)*1.8,1]); axis square;% 设定坐标轴显示范围
hold on;box on;
gh1=plot([0,L1*exp(i*(x(1)-pi/2))],'r-');% 显示上面的摆
set(gh1,'linewidth',2,'markersize',6,'marker','o');% 设置线宽和标记符号
gh2=plot([L1*exp(i*(x(1)-pi/2)),L1*exp(i*(x(1)-pi/2))+L2*exp(i*(x(2)-pi/2))],'
b-');%  显示下面的摆
set(gh2,'linewidth',2,'markersize',6,'marker','o');% 设置线宽和标记符号
for k=2:size(x,1);
    C1=[0,L1*exp(i*(x(k,1)-pi/2))]; %获得上侧摆位置数据

C2=[L1*exp(i*(x(k,1)-pi/2)),L1*exp(i*(x(k,1)-pi/2))+L2*exp(i*(x(k,2)-pi/2))]; %
获得下侧摆位置数据
    set(gh1,'xdata',real(C1),'ydata',imag(C1));% 更新上侧摆的位置
    set(gh2,'xdata',real(C2),'ydata',imag(C2));% 更新下侧摆的位置
    title(['t= ',num2str(t(k))],'fontsize',12);% 更新题注中时间数值
    pause(0.1);% 暂停
end
figure;
subplot(231);plot(t,x(:,1));title('t-\theta_1');xlabel('t');ylabel('\theta_1')
;%摆的位置数据
subplot(232);plot(t,x(:,2));title('t-\theta_2');xlabel('t');ylabel('\theta_2')
;%摆的位置数据
subplot(233);plot(t,x(:,3));title('t-\omega_1');xlabel('t');ylabel('\omega_1')
;%摆的速度数据
subplot(234);plot(t,x(:,4));title('t-\omega_2');xlabel('t');ylabel('\omega_2')
;%摆的速度数据
subplot(235);plot(x(:,1),x(:,3));title('\theta_1-\omega_1');xlabel('\theta_1')
;ylabel('\omega_1');%
subplot(236);plot(x(:,2),x(:,4));title('\theta_2-\omega_2');xlabel('\theta_2')
;ylabel('\omega_2');%
```

动画结束后的画面如图 19.15 所示。

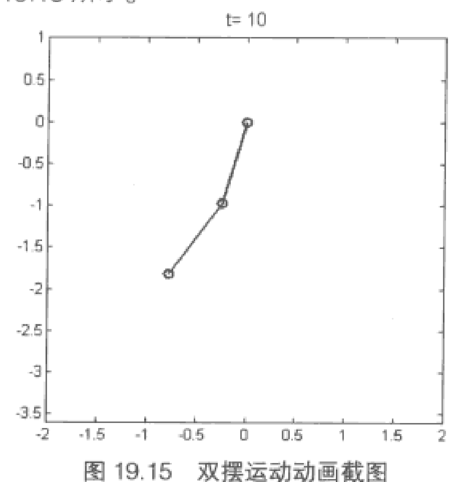


图 19.15 双摆运动动画截图

相应的位置、速度关系曲线如图 19.16 所示。

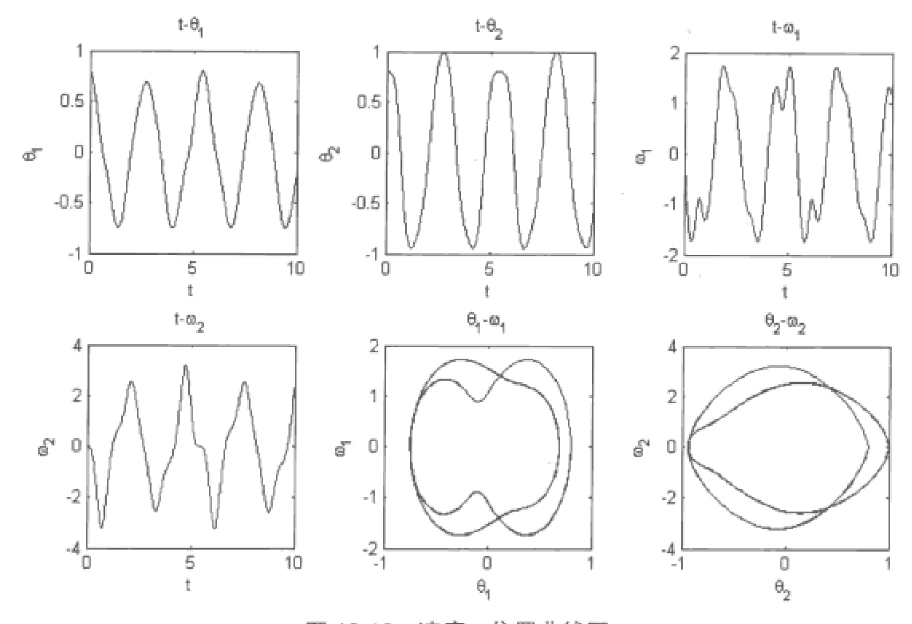


图 19.16 速度、位置曲线图

总而言之，对于程序的正确性需要进行多方面充分的验证以求保证。在一些研究中还需要进行数值计算结果和科学实验结果的比对，二者吻合可以对所做的结论进行双重验证。此外他人的计算结果也可以作为比较的对象。

19.6 算法优化

作者认为编程的第一原则是正确地写出目标程序，然后对程序修饰。追求高效的算法可以加快对于问题的研究进程，但算法优化的过程可能会降低程序的可读性。算法优化包括两方面的内容：一是算法流程的优化，二是程序段的优化。前面介绍的程序加速方法可以用于程序段的优化，以提

高程序的执行速度。在循环结构中，与循环指标无关的量可以转到循环前面进行计算。同时还要选择执行速度快的函数用于计算中。本节主要讨论算法流程的优化。

对于某一问题可能存在多种解决方案，此时需要进行选择，选出其中最快的判断方法，或者称为捷径。这样可以节省程序的编写时间，同时提高程序的执行速度。对于程序流程的优化需要研究者对问题有充分的了解，这可能需要长时间的思考和探索。有时多个人分别独立解决同一个问题，可能会有不同角度的认知，从而产生相对简单、巧妙的思路。

这里以前面 19.4 节中判断 C 和 D 两点在直线两侧的问题为例，考虑其他的等价方法。过 AB 两点的直线方程可以写为：

$$f(x,y)=(\sin\beta_1-\sin\alpha_1)x-(\cos\beta_1-\cos\alpha_1)y+\sin\alpha_1\cos\beta_1-\cos\alpha_1\sin\beta_1=0 \quad (19\text{-}10)$$

或

$$f(x,y)=(\sin\beta_1-\sin\alpha_1)x-(\cos\beta_1-\cos\alpha_1)y+\sin(\alpha_1-\beta_1)=0 \quad (19\text{-}11)$$

在平面上的各点可以通过计算 $f(x,y)$ 来确定点和直线的位置关系：直线左侧、右侧以及直线上，它们对应于 $f(x,y)$ 的正、负以及 0。把 C 点和 D 点的坐标带入上述表达式中就可以进行判断了。利用上面公式所提供的思路，写出以下的程序。

```
N=1000000;                                  % 实验总次数
intersect=0;                                  % 统计相交次数
fxy=inline('[sin(b1)-sin(a1)]*x-[cos(b1)-cos(a1)]*y+sin(a1-b1)','x','y','a1','
b1');
for k=1:N;
   alpha1=rand*pi*2;                          % 随机产生一个复角
   alpha2=rand*pi*2;                          % 随机产生一个复角
   beta1=rand*pi*2;                           % 随机产生一个复角
   beta2=rand*pi*2;                           % 随机产生一个复角
   F1=fxy(cos(alpha2),sin(alpha2),alpha1,beta1);  % 计算位置关系式
   F2=fxy(cos(beta2),sin(beta2),alpha1,beta1);    % 计算位置关系式
   intersect=intersect+(F1*F2<=0);               % 如果相交，则相交次数累加 1
end
format long                                   % 输出更多数位
p=intersect/N                                  % 输出弦相交的几率
```

输出结果为：

```
p =   0.333442000000000
```

采用 cputime 函数计算程序执行时间，19.4 节中相应程序所需时间为 0.281 秒，而本节程序所需时间为 987.75 秒。可见二者差异很大，多次调用函数 inline 会花费很多时间。

在 19.4 节中关于计算弦相交几率的程序还可以进一步改为如下的等价形式。

```
N=1000000;        % 实验总次数
alpha1=rand(1,N);  % 随机产生一个复角
alpha2=rand(1,N);  % 随机产生一个复角
beta1=rand(1,N);   % 随机产生一个复角
beta2=rand(1,N);   % 随机产生一个复角
F1=[alpha2-alpha1].*[alpha2-beta1];  % 计算位置关系式
F2=[beta2-alpha1].*[beta2-beta1];    % 计算位置关系式
intersect=length(find(F1.*F2<=0))    % 统计相交次数
format long                     % 输出更多数位
p=intersect/N                     % 输出弦相交的几率
```



上述程序中未使用循环，同时省去了2*pi这个公因子。程序执行的时间为0.1563秒，可见程序进一步加快了。关于程序计算时间的统计决定于计算机性能及程序所处的系统环境，如内存和CPU占用情况。虽然具体时间数值可能因计算机而有所不同，但不同时间的大小关系是不变的。

上述程序输出的计算结果为：

```
p =   0.333018000000000
```

从前面的例子可以看出程序的算法结果可以使程序的计算时间、繁简程度有很大不同。一般地，程序越简单执行效率越高。

19.7 小结

本章主要介绍了建模方面的基础知识。了解这方面的知识对解决实际问题、把握整体脉络非常重要。其中一个重要环节就是把实际问题转化为数学化的语言和程序化的结构。通过逻辑思维得到问题的抽象等价描述，进而利用MATLAB软件进行仿真和数值模拟。首先介绍了从科学和工程问题中抽象数学模型的方法，这是建模的第一个环节。对于离散模型的程序化方法是把实际问题转化为程序的一个中转环节。对于离散模型，计算机可以很好地模拟。在数学模型的基础上建立程序结构对于建立程序具有指导意义，本章相应地给出了实例求解方法分析。对于计算所得结果的验证是使程序普适化的一个必要环节，通过不同参数、不同角度的检测可以确认程序是否通用。最后介绍了程序优化的方法，用户可以从流程图和程序段两个方面来优化已经存在的程序，通过优化可以提高程序的执行效率和执行速度。

◆ ◆ ◆

第 20 章 混沌现象

本章包括

- ◆ **离散混沌** 介绍几种典型的映射和简单函数构成的映射关系对应的分岔图形的绘制。
- ◆ **微分方程中的分岔和混沌行为** 介绍微分方程中存在的混沌现象的程序模拟。
- ◆ **混沌吸引子** 介绍相图、混沌吸引子以及庞加莱截面的绘制方法。
- ◆ **Lyapunov 指数** 介绍映射和微分方程的李雅普诺夫指数的计算。

混沌是指发生在确定性系统中的貌似随机的不规则运动。一个确定性理论描述的系统，其行为却表现出不确定性——不可重复、不可预测，这就是混沌现象。大量研究表明，混沌是非线性动力系统的固有特性，它是非线性系统普遍存在的现象。牛顿确定性理论能够充分处理的多为线性系统，而线性系统大多是由非线性系统通过某些简化而获得的。因此在现实生活、工程技术以及科学问题中，混沌现象是无处不在的。通过数值的方法研究混沌现象是比较方便的。本章主要介绍利用 MATLAB 模拟一些混沌现象。

20.1 离散混沌

早在 1972 年 12 月 29 日，美国麻省理工学院教授、混沌学主要开创人之一洛伦兹在美国科学发展学会第 139 次会议上发表了题为《蝴蝶效应》的论文，提出一个看似荒谬的论断：在巴西一只蝴蝶翅膀的拍打能够在美国得克萨斯州产生一个龙卷风，并由此提出了天气的不可准确预报性。时至今日，这一论断仍为人津津乐道，更重要的是它激发了人们对混沌现象研究的浓厚兴趣。伴随计算机技术的飞速发展，混沌学已经逐渐成为一门影响深远、发展迅速的前沿科学。目前每年有大量相关科学论文发表。

一般情况下，如果一个接近实际而没有内在随机性的模型仍然具有貌似随机的行为，就可以称这个真实物理系统是混沌的。一个随时间确定性变化或具有微弱随机性的变化系统，称为动力系统，它的状态可由一个或几个物理量确定。而在一些动力系统中，两个几乎完全一致的状态经过足够长时间后会变得完全不一致，恰如从长序列中随机选取的两个状态那样，这种系统被称为混沌系统。而对初始条件的敏感依赖性也可看做是混沌现象的一个定义。

混沌现象来自于非线性动力系统，而动力系统又描述的是任意随时间发展变化的过程，并且这样的系统可见于生活的不同方面。举例来说，生态学家对某物种的长期性态感兴趣，给定一些观察到的或实验得到的变量（如捕食者个数、气候的恶劣性、食物的可获性等数据），建立数学模型来描述群体的增减。如果用 Pn 表示 n 代后该物种极限数目的百分比，则著名的“罗杰斯蒂映射” Pn+1=kP(1-Pn)（其中 k 是依赖于生态条件的常数，n+1 是脚标）可以用于在给定 P0 和 k 条件下，预报群体数的长期性态。如果将常数 k 处理成可变的参数 k，则当 k 值增大到一定值后，“罗杰斯蒂映射”所构成的动力系统就进入混沌状态。稍后将给出该映射的程序实现。

在非线性科学中，“混沌”一词的含义和本意相似但又不完全相同。非线性科学中的混沌现象

指的是一种确定的但不可预测的运动状态。它的外在表现和纯粹的随机运动很相似，即都不可预测。但和随机运动不同的是，混沌运动在动力学上是确定的，它的不可预测性来源于运动的不稳定性。或者说混沌系统对无限小的初值变动和微扰也具于敏感性，无论多小的扰动在长时间以后，也会使系统彻底偏离原来的演化方向。混沌现象是自然界中的普遍现象，天气变化就是一个典型的混沌运动。混沌系统具有 3 个关键要素：

- 对初始条件的敏感依赖性。
- 临界水平，这里是非线性事件的发生点。
- 分形维，它表明有序和无序的统一。

混沌系统经常是自反馈系统，出来的东西会回去经过变换出来，循环往复，没完没了，任何初始值的微小差别都会按指数放大，因此导致系统内在地不可长期预测。混沌确定系统是庞加莱在研究三体问题时首次发现的。

本节介绍一些典型映射关系，如罗杰斯蒂映射、埃农映射、帐篷映射以及肯特映射等，这些映射根据参数的不同可以表现出分岔、混沌等现象。下面举例来说明这些映射的模拟。

20.1.1 罗杰斯蒂映射

罗杰斯蒂（logistic）映射的数学表达式为：

$$x_{n+1}=ax_n\left(1-x_n\right) \tag{20-1}$$

这里 a 是参数，随着参数 a 的取值不同上面的迭代公式可以表现出不同的行为。罗杰斯蒂映射相应的实现程序如下：

```
an=linspace(3.1,3.99,400);% 生成参数 a 的离散采样值
hold on; box on; axis([min(an),max(an),-1,2]); % 设置坐标轴范围
N=1000; % 设置迭代次数
xn=zeros(1,N); % 预置 x 的变量占用的空间
for a=an;
   x=rand; % 随机赋初值
   for k=1:20;
       x=a*x*(1-x); % 预迭代 20 次，以达到相应的混沌状态
   end
   for k=1:N;
       x=a*x*(1-x); % 按预定次数 N 多次迭代，并把这部分数据作为绘图数据
       xn(k)=x; % 记录迭代结果到数组 xn 中
   end
   plot(a*ones(1,N),xn,'k.','markersize',1); % 绘图
end
```

执行上述程序后可以得到如图 20.1 所示的图形。

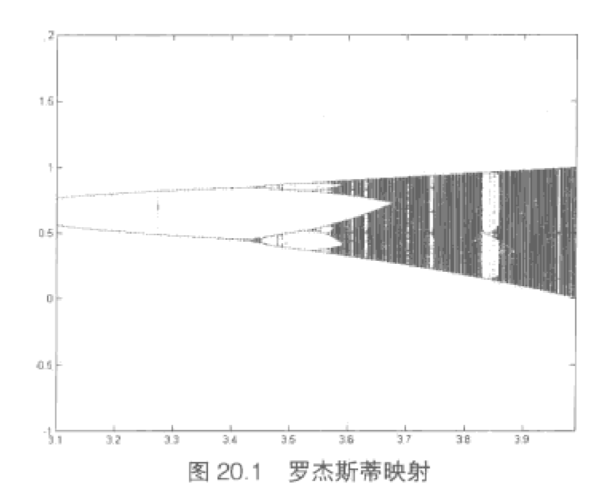


图 20.1　罗杰斯蒂映射



在上面的程序中，先进行 20 次循环以使迭代达到一个混沌状态。相应的循环次数用户可以根据实际情况确定。绘图时利用点符号“.”且把点的大小设置为 1，这样所得到的图形是一点点画出来的。

可见在罗杰斯蒂映射中随着参数取值不同可以得到分岔、混沌行为。开始时罗杰斯蒂映射是有两个分岔，到 3.45 左右出现 4 个分岔，在参数 a 等于 3.6 左右时出现混沌状态。对于参数 a 的其他范围用户可以利用上述程序结构进行研究。

20.1.2　埃农映射

埃农（Henon）映射（有的书上也称为平方映射）的数学表达式为：

$$\text{Henon1}:\begin{cases} x_{n+1} = y_n + 1 - a x_n^2 \\ y_{n+1} = b x_n \end{cases} \tag{20-2}$$

相应的实现程序为，这里取参数 $b=0.3$，$a\in[0,1.4]$。

```
b=0.3; % 设置参数 b 的数值
N=400; % 设置迭代次数
an=ones(1,N); % 生成一个全 1 数组
xn=zeros(1,N); % 预置 xn 变量所占用的空间
hold on;box on;
x=0; % 迭代初值
y=0; % 迭代初值
for a=0:0.001:1.4; % 对参数 a 离散取值
    for k=1:N; % 预迭代 N 次以达到混沌状态
        xm=x; % xm 为 x 的迭代之前的数值
        ym=y; % ym 为 y 的迭代之前的数值
        x=ym+1-a*xm.*xm; % 迭代计算
        y=b*xm; % 迭代计算
    end
    xn(1)=x; % 记录初值
    for n=2:N;
        xm=x; % xm 为 x 的迭代之前的数值
```

```
        ym=y; % ym为y的迭代之前的数值
        x=ym+1-a*xm.*xm; % 迭代计算
        y=b*xm; % 迭代计算
        xn(n)=x;% 记录数值
    end
    plot(an*a,xn,'k.','markersize',1); % 绘制点图
end
xlim([0,a]);% 设置坐标轴范围
title(['Henon Bifurcation, a=0~',num2str(a),', b=0.3']); % 添加图形题目
```



用户可以选择参数 a 和 b 的其他取值情况进行研究，其中参数 a 的最大取值是 1.4，大于该值迭代将会发散。

执行上述程序所得图形如图 20.2 所示。

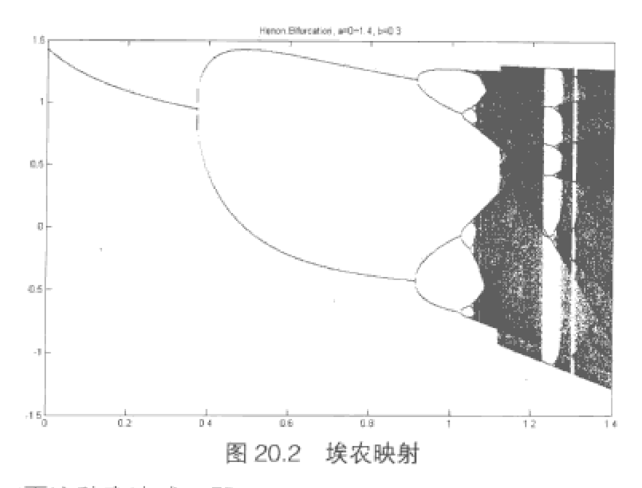


图 20.2 埃农映射

埃农映射还存在下面这种表达式，即：

$$\begin{cases} x_{n+1} = a - x_n^2 + by_n \\ y_{n+1} = x_n \end{cases} \quad (20\text{-}3)$$

把表达式代入到前面的程序，可以得到如图 20.3 所示的图形。

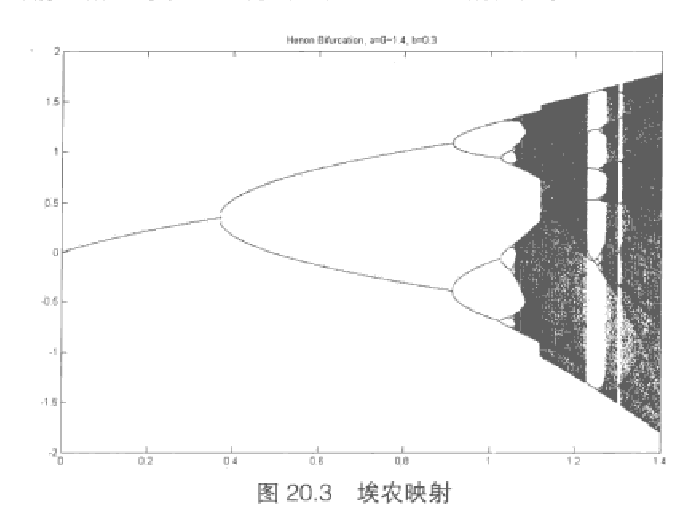


图 20.3 埃农映射

20.1.3 帐篷映射

帐篷映射的数学定义是：

$$x_{n+1} = a - (1+a)|x_n| \tag{20-4}$$

根据公式（20-4）可以写出帐篷映射的计算程序，如下：

```
N=200; % 设置迭代次数
xp=zeros(1,N); % 为绘图数据预设空间
Aa=ones(1,N); % 生成全 1 向量
hold on;box on;
x=0.34; % 设置初值
for a=0:.001:1;
   for n=1:N;
      x=a-(1+a)*abs(x); % 预迭代使之进入混沌状态
   end
   for k=1:N;
      x=a-(a+1)*abs(x); % 迭代生成绘图数据
      xp(k)=x; % 记录绘图数据
   end
   plot(Aa*a,xp,'k.','markersize',1); % 绘图
end
xlim([0,a]); % 设置坐标轴范围
xlabel('\ita','Fontsize',22,'Fontname','Times new roman'); % X 轴标注
```



这里参数 a 取值范围在 $[0,1]$ 内存在混沌现象。

执行上述程序输出图形如图 20.4 所示。

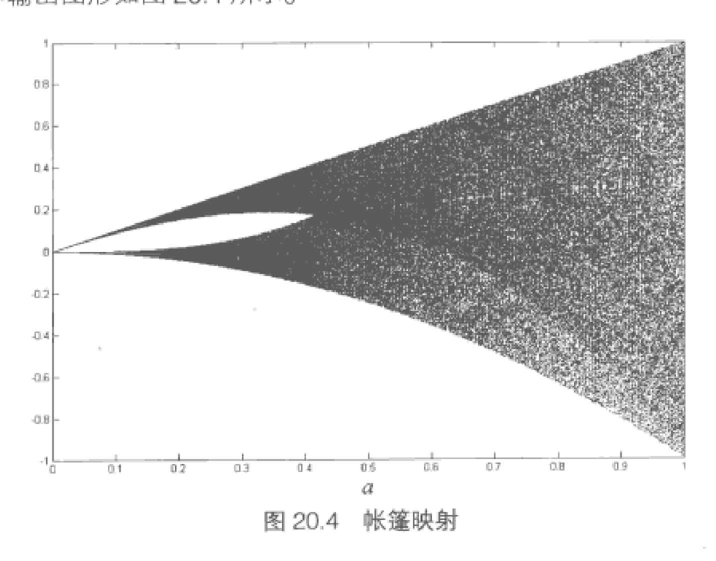


图 20.4 帐篷映射

20.1.4 肯特映射

肯特（Kent）映射的数学表达式为：

$$x_{n+1}=\begin{cases}\dfrac{x_n}{a}, & 0<x\le a\\ \dfrac{1-x_n}{1-a}, & a<x<1\end{cases} \tag{20-5}$$

根据公式（20-5）可以写出肯特映射的实现程序，具体内容如下：

```
N=200;          % 设置迭代次数
xp=zeros(1,N);  % 为绘图数据预设空间
Aa=ones(1,N);  % 生成全 1 向量
x=0.36;         % 设置初值
hold on;box on;
for a=0.01:.001:0.5;
   for k=1:N;
       x=x/a*[x<=a]+(1-x)/(1-a)*[x>a];  % 预迭代使之进入混沌状态
   end
   for n=1:N;
       x=x/a*[x<=a]+(1-x)/(1-a)*[x>a];  % 迭代生成绘图数据
       xp(n)=x; % 记录绘图数据
   end
   plot(Aa*a,xp,'k.','markersize',1);     % 绘图
end
xlabel('\ita','Fontsize',22,'Fontname','Times new roman');    % X 轴标注
```



在肯特映射中，参数 a 在 $(0,0.5]$ 范围。程序执行后输出图形如图 20.5 所示。

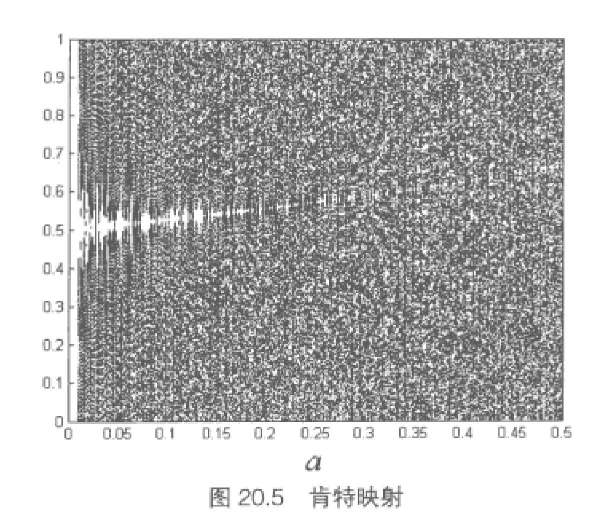


图 20.5 肯特映射

20.1.5 Lozi 映射

Lozi 映射的定义是：

$$\begin{cases}x_{n+1}=1-p|x_n|+y_n\\ y_{n+1}=qx_n\end{cases} \tag{20-6}$$

下面考虑计算参数 p 和 q 取特定值时的图形，如下一段程序用来计算当 $p=1.75$，$q=0.33$ 时迭代点形成的图形。

```
N=10000;      % 设置迭代次数
x=zeros(1,N);   % 初始化变量 x
y=x;      % 初始化变量 y
q=0.33;   % 对参数 q 赋值
p=1.75;   % 对参数 p 赋值
axes('Position',[0.2,0.2,0.6,0.6]);  % 设置坐标轴位置
for n=1:N-1;
   x(n+1)=1-1.75*abs(x(n))+y(n);  % 迭代计算
   y(n+1)=q*x(n);          % 迭代计算
end
plot(x,y,'.','markersize',1);  % 绘图
title(['{\itp}=',num2str(p),',{\itq}=',num2str(q)],'Fontsize',16,'Fontname','T
imes new roman');%标注图题
xlabel('\itx','Fontsize',16,'Fontname','Times new roman');  % X 轴标注
ylabel('\ity','Fontsize',16,'Fontname','Times new roman');   % Y 轴标注
```

利用公式（20-6）定义的迭代式进行迭代，这里初值选为 $x_0=0$，$y_0=0$。依次记录下迭代过程的数值并画图。其中对数据画图时采用点把数据点画出来，而不是用连线的方式画图。输出图形如图 20.6 所示，可见迭代过程得到的轨迹是斜线形的，且外面的折线较粗。

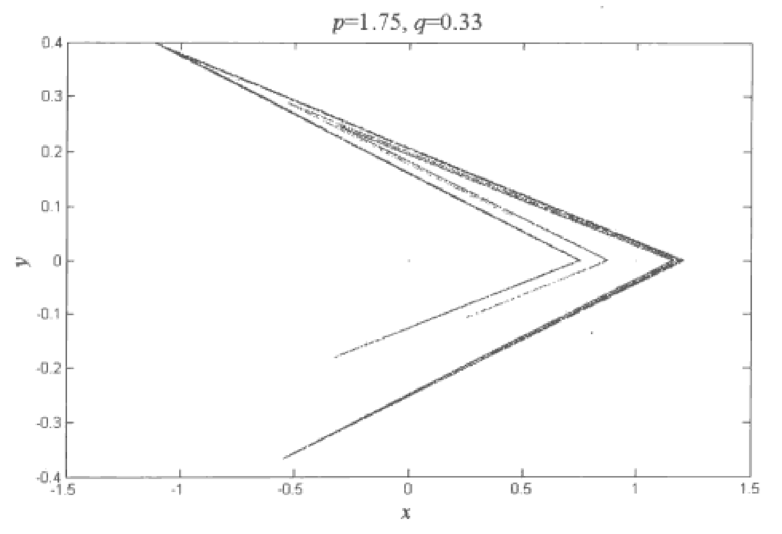


图 20.6　Lozi 映射迭代点 $p=1.75$，$q=0.33$ 形成的图形

下面一段程序用来实现当参数 $q=0.23$ 且 $p\in[0,1.7]$ 时的 Lozi 映射图。

```
N=150;       % 设置迭代次数
xn=zeros(1,N); % 初始化记录 x 数值的向量
Pp=ones(1,N); % 生成全 1 向量
figure;axes('Position',[0.2,0.2,0.6,0.6]);% 设置坐标轴位置
hold on;box on;
x=0.34; % 设置初值
y=0;   % 设置初值
q=0.23; % 对参数 q 赋值
for p=0:.001:1.7;
```

```
    for n=1:N;
        xt=x; % xt 是迭代前 x 的数值
        yt=y; % yt 是迭代前 y 的数值
        x=1-p*abs(xt)+yt;% 预迭代使之进入混沌状态
        y=q*xt;% 预迭代使之进入混沌状态
    end
    xp(1)=x;%记录初值
    for k=2:N;
        xt=x; % xt 是迭代前 x 的数值
        yt=y; % yt 是迭代前 y 的数值
        x=1-p*abs(xt)+yt;% 计算迭代过程数值
        y=q*xt;    % 计算迭代过程数值
        xp(k)=x;    % 记录绘图数据
    end
    plot(Pp*p,xp,'.','markersize',1); % 绘图
end
xlim([0,p]);        % 设置坐标轴范围
xlabel('\itp','Fontsize',16,'Fontname','Times new roman');  % X 轴标注
ylabel('\itx','Fontsize',16,'Fontname','Times new roman');  % Y 轴标注
title(['{\itq}=',num2str(q)],'Fontsize',16,'Fontname','Times new roman'); % 标注
图题
```

分别取不同的 p 值进行迭代计算，在记录迭代数据前先进行 N 次预迭代以稳定状态，然后开始记录数据用于绘图。执行上述程序输出图形如图 20.7 所示，可见在 p 取不同数值时，x 可能出现分岔、混沌现象。

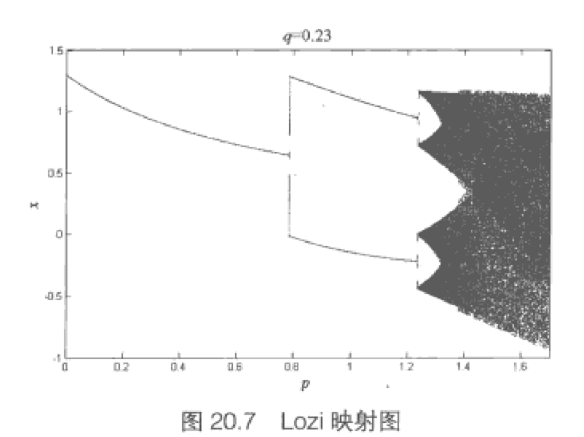


图 20.7　Lozi 映射图

20.1.6　Ushiki 映射

Ushiki 映射的数学定义为：

$$\begin{cases} x_{n+1} = (a - x_n - by_n)x_n \\ y_{n+1} = (a - cx_n - y_n)y_n \end{cases} \tag{20-7}$$

取 $b = 0.1$，$c = 0.2$ 且 $a \in [2.5, 3.8]$ 时，绘制相应的映射图形的程序如下：

```
N=20000; % 设置迭代次数
x=zeros(1,N);y=x; % 对 x 和 y 预定义初值
x(1)=0.32;y(1)=0.32; % 定义初值
a=3.7; % 对参数 a 赋值
```

```
b=0.1; % 对参数 b 赋值
c=0.2; % 对参数 c 赋值
for n=1:N-1;
   x(n+1)=(a-x(n)-b*y(n))*x(n); % 迭代计算
   y(n+1)=(a-c*x(n)-y(n))*y(n); % 迭代计算
end
N=100; % 设置迭代次数
subplot(131);plot(x,y,'.','markersize',1); % 绘图
xlabel('\itx','Fontsize',16,'Fontname','Times new roman'); % X 轴标注
ylabel('\ity','Fontsize',16,'Fontname','Times new roman'); % Y 轴标注
title(['{\ita}=',num2str(a),', {\itb}=',num2str(b),',
{\itc}=',num2str(c)],'Fontsize',16,'Fontname','Times new roman');% 标注图题
s(1)=subplot(132);hold on;box on;% 生成坐标轴
xlabel(s,'\ita','Fontsize',16,'Fontname','Times new roman'); % X 轴标注
ylabel(s,'\itx','Fontsize',16,'Fontname','Times new roman'); % Y 轴标注
title(s,['{\itb}=',num2str(b),',{\itc}=',num2str(c)],'Fontsize',16,'Fontname',
'Times new roman');
s(2)=subplot(133);hold on;box on;% 生成坐标轴
xa=zeros(1,N); % 初始化记录 x 数值的向量
ya=zeros(1,N); % 初始化记录 y 数值的向量
Aa=ones(1,N); % 生成全 1 向量
x=0.32; % 设置初值
y=0.32; % 设置初值
for a=2.5:.0005:3.8;
   x=0.2; % 设置每步迭代的初值
   y=0.1; % 设置每步迭代的初值
   for n=1:60;
       xt=x; % xt 是迭代前 x 的数值
       yt=y; % yt 是迭代前 y 的数值
       x=[a-xt-b*yt]*xt;% 预迭代使之进入混沌状态
       y=[a-c*xt-yt]*yt;% 预迭代使之进入混沌状态
   end
   xa(1)=x;%记录初值
   ya(1)=y;%记录初值
   for k=2:N;
       xt=x; % xt 是迭代前 x 的数值
       yt=y; % yt 是迭代前 y 的数值
       x=[a-xt-b*yt]*xt;% 计算迭代数值
       y=[a-c*xt-yt]*yt;% 计算迭代数值
       xa(k)=x; % 记录绘图数据
       ya(k)=y; % 记录绘图数据
   end
   plot(s(1),Aa*a,xa,'k.','markersize',1); % 绘图
   plot(s(2),Aa*a,ya,'r.','markersize',1); % 绘图
end
set(s,'XLim',[2.5,a]); % 设置坐标轴范围
xlabel('\ita','Fontsize',16,'Fontname','Times new roman'); % X 轴标注
ylabel('\itx','Fontsize',16,'Fontname','Times new roman'); % Y 轴标注
title(['{\itb}=',num2str(b),',
{\itc}=',num2str(c)],'Fontsize',16,'Fontname','Times new roman');%标注图题
```



语句“axes(s(1)); plot(Aa*a,xa,'k.','markersize',1);”和“plot(s(1),Aa*a,xa,'k.','markersize',1);”作用等价，但是执行速度差异很大，前者非常慢。

上述程序所得结果如图 20.8 所示。其中左图表示给定 a，b 和 c 三个参数时迭代所得点坐标对应的图，中间和右侧的图形分别是 x_n 和 y_n 对参数 a 的图形。可以发现 x_n 和 y_n 随着参数 a 的不同出现了分岔和混沌行为。

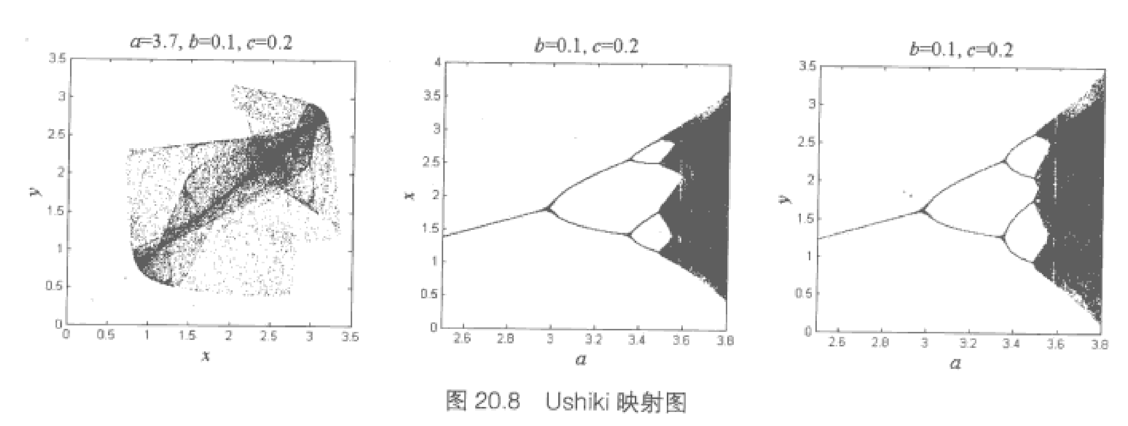


图 20.8 Ushiki 映射图

当参数 b 和 c 在一定范围内取值时对应的 Ushiki 映射图如图 20.9 所示（相应计算程序保存为光盘中\Ch20 文件夹下的 Ushiki2_maping.m 文件），相应参数取值在图 20.9 中已经标出。可以看出，参数取不同数值对于混沌行为影响很大。

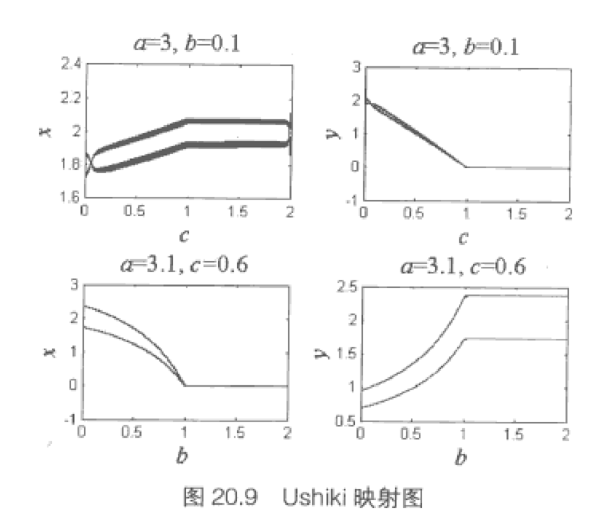


图 20.9 Ushiki 映射图

20.1.7 三个迭代式形成的映射关系

下面介绍 3 个迭代式形成的映射关系：

$$T_1: x_{n+1} = a\sin(\pi x_n) \tag{20-8}$$

$$T_2: x_{n+1} = 1 - a x_n^2 \tag{20-9}$$

$$T_3: x_{n+1} = (k+1)x_n - k x_n^2 \tag{20-10}$$

第 1 个映射 T_1 的实现程序如下：

```
N=300;  % 取样点数
starx=0.1; % 设置初值
Z=zeros(1,N); % 生成空的数组
```

```
Aa=ones(1,N);% 生成全 1 向量
figure;hold on;box on; % 生成空的图形窗口
for a=0.5:0.0001:1;   % 参数离散取值
   x=starx; % 设置初值
   for k=1:400;
      x=a*sin(pi*x);  % 预迭代使之进入混沌状态
   end
   for k=1:N;
      x=a*sin(pi*x); % 迭代过程数值
      Z(k)=a+x*i; % 利用复数记录坐标点
   end
   plot(Z,'m.','markersize',1); % 绘图
end
xlim([0.5,a]);% 设置坐标轴范围
xlabel('\ita','Fontsize',16,'Fontname','Times new roman'); % X 轴标注
```

输出的图形如图 20.10 左图所示，右图是映射 $x_{n+1}=a\cos(\pi x_n)$ 对应的图形（程序与上面程序相似，这里不再说明，可以查阅光盘中\Ch20 文件夹下的 fun_map3.m 文件）。

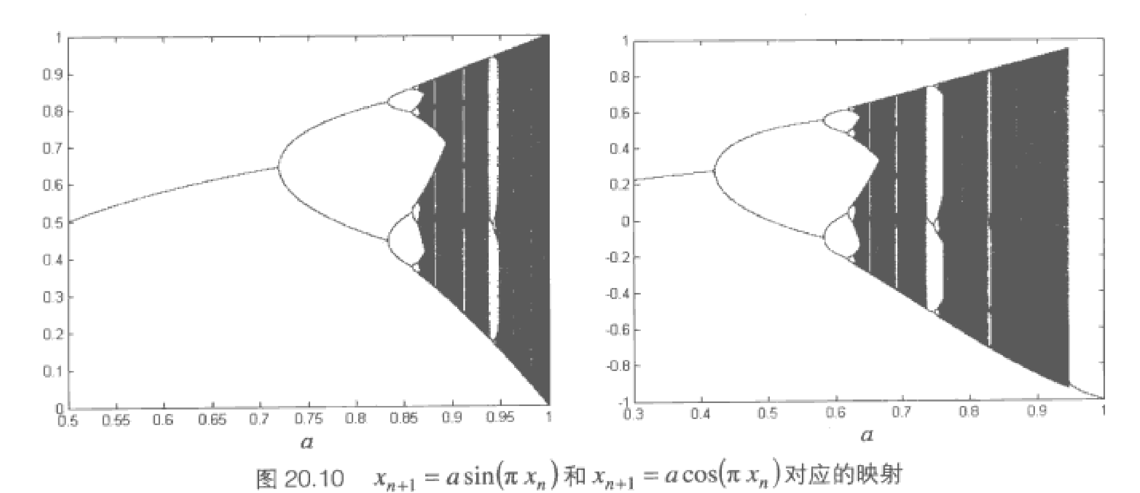


图 20.10　$x_{n+1}=a\sin(\pi x_n)$ 和 $x_{n+1}=a\cos(\pi x_n)$ 对应的映射

第 2 个映射 T_2 对应的程序如下：

```
N=300;    % 取样点数
starx=0.1;  % 设置初值
Z=zeros(1,N);  % 生成空的数组
Aa=ones(1,N); % 生成全 1 向量
figure;hold on;box on;  % 生成空的图形窗口
for a=0:0.0001:1.9;   % 参数离散取值
   x=starx; % 设置初值
   for k=1:N;
      x=1-a*x^2;  % 预迭代使之进入混沌状态
   end
   for k=1:N;
      x=1-a*x^2; % 迭代过程数值
      Z(k)=a+x*i; % 利用复数记录坐标点
   end
   plot(Z,'k.','markersize',1); % 绘图
end
xlim([0,a]);ylim([-1,1.1]);% 设置坐标轴范围
```

```
xlabel('\ita','Fontsize',16,'Fontname','Times new roman'); % X轴标注
```

利用公式（20-9）进行预迭代之后，再记录数据用于绘图。其中对于每个参数 a 的数值，都是用相同的初始数值作为初始条件进行迭代的。上述程序输出的图形如图 20.11 所示，可见该映射的图形呈现出倍周期、4 周期、8 周期和混沌行为。

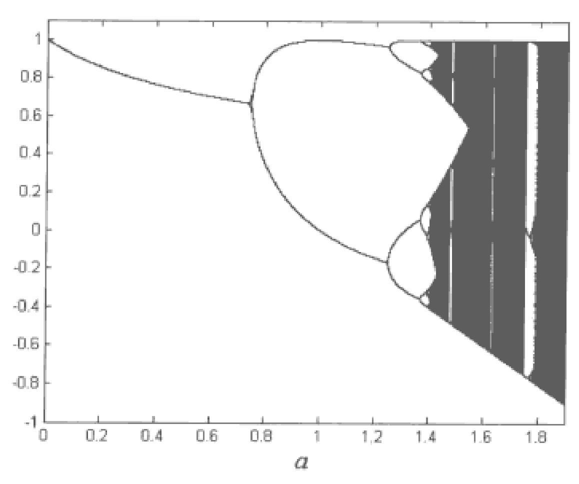


图 20.11　$x_{n+1}=1-ax_n^2$ 对应的映射

（3）第 3 个映射 T_3 对应的程序如下：

```
x=0.2; % 迭代初值
N=400; % 设置迭代次数
zp=zeros(1,N); % 生成变量的数组预留空间
hold on;box on; % 生成空的坐标轴
for k=1:.001:3;
   for p=1:80;
       x=-k*x^2+(k+1)*x; % 预迭代进入混沌状态
   end
   for p=1:N;
       x=-k*x^2+(k+1)*x; % 迭代计算
       zp(p)=k+x*i; % 以复数的形式记录(p,xn)点
   end
   plot(zp,'k.','markersize',1); % 绘图
end
xlim([1,k]); % 设置坐标轴范围
xlabel('\itk','Fontsize',16,'Fontname','Times new roman'); % X轴标注
```

利用公式（20-10）进行预迭代之后，再记录数据用于绘图。其中对于每个参数 k 的数值，都是使用前一个 k 值迭代后 x 的数值作为初始条件进行迭代的，而最开始 x 的初始值设为 0.2，最后给出相应的分岔图形。执行上述程序输出图形如图 20.12 所示，该映射在不同 k 值下呈现出倍周期、4 周期、8 周期和混沌行为。

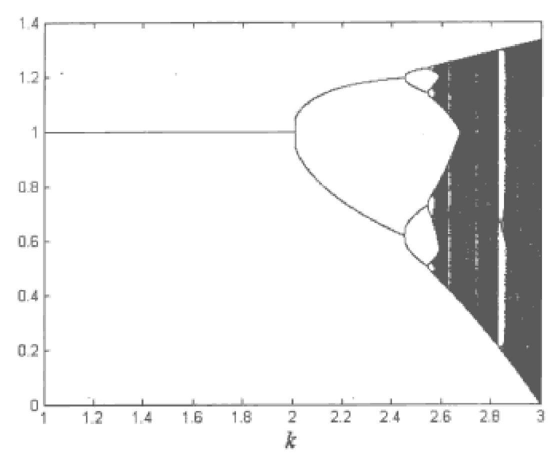


图 20.12　第 3 个映射 $x_{n+1}=(k+1)x_n-kx_n^2$ 对应的图形

20.1.8　双混沌图形

前面介绍的混沌图形一般是横轴变化的参数，纵轴是迭代值 x_n。如果横纵轴均用迭代值表示，相应的图形会是什么样子呢？这样的图形就是双混沌图形，下面给出双混沌图的定义。记 T_1 和 T_2 表示两个映射关系分别生成 x_n 和 y_n，中间变量 R 和 A 定义为：

$$R=x_{n+1}+y_{n+1},\quad A=x_n+y_n \tag{20-11}$$

$$X=R\cos(K\,\pi\,A),\quad Y=R\sin(K\,\pi\,A) \tag{20-12}$$

其中 X 和 Y 表示绘图的坐标点。

下面举例说明双混沌图形的绘制，用到的映射关系是 $x_{n+1}=a\sin(\pi\,x_n)$ 和 $x_{n+1}=a\cos(\pi\,x_n)$，实现程序如下：

```
N=300;  % 取样点数
starx=0.1; % 设置初值
Z1=zeros(1,N); % 生成空的数组
Z2=zeros(1,N); % 生成空的数组
K=pi; % 设置参数 K 的数值
Aa=ones(1,N);% 生成全 1 向量
figure;s(1)=subplot(121);hold on;box on; % 生成空的坐标轴
xlabel('\itx','Fontsize',16,'Fontname','Times new roman'); % X 轴标注
ylabel('\ity','Fontsize',16,'Fontname','Times new roman'); % Y 轴标注
s(2)=subplot(122);hold on;box on; % 生成空的坐标轴
for a=0.5:0.0001:1;   % 参数离散取值
   x=rand; % 设置初值
   y=rand; % 设置初值
   for k=1:400;
      x=a*sin(pi*x);  % 预迭代使之进入混沌状态
      y=a*cos(pi*y);  % 预迭代使之进入混沌状态
   end
   for k=1:N;
      A=x+y; % 计算参数 A
      x=a*sin(pi*x); % 计算迭代过程数值
      y=a*cos(pi*y); % 计算迭代过程数值
      R=x+y;% 计算参数 R
      Z1(k)=x+y*i; % 利用复数记录坐标点
```

```
      Z2(k)=R*exp(i*K*A*pi); % 利用复数记录坐标点
   end
   plot(s(1),Z1,'m.','markersize',1); % 绘图
   plot(s(2),Z2,'r.','markersize',1); % 绘图
end
set(s,'XLim',[0.5,a]);axis(s,'square'); % 设置坐标轴范围和形状
xlabel('\itX','Fontsize',16,'Fontname','Times new roman'); % X 轴标注
ylabel('\itY','Fontsize',16,'Fontname','Times new roman'); % Y 轴标注
```

根据映射关系进行预迭代之后，再记录数据用于绘图。其中对于每个参数 a 的数值，都是用服从均匀分布的随机数作为初始条件进行迭代的，最后给出双映射对应的分岔图形。执行上述程序输出的图形如图 20.13 所示，其中左图是点(x_n, y_n)对应的图形，右图是点(X,Y)对应的图形，可见双映射的图案是在分岔图上面“蒙”上了一层模糊的随机点。

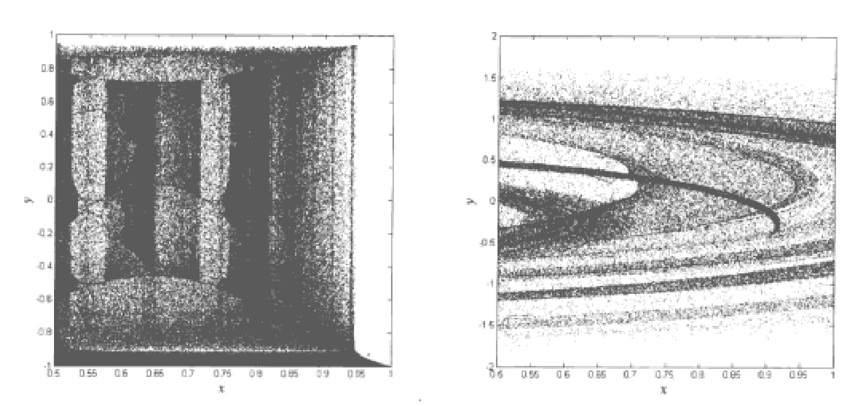


图 20.13 双混沌图形

20.1.9 标准映射

标准映射的定义为：

$$\begin{cases} y_{n+1} = y_n + K\sin x_n \\ x_{n+1} = x_n + y_{n+1} \end{cases} \tag{20-13}$$

其中参数K，通过绘制(x_n, y_n)可以得到标准映射的相图，标准映射相图的程序如下：

```
K=1/2; % 设置参数 K 的数值
N=800; % 迭代次数
M=10; % 整体迭代重数
Zz=zeros(1,N); % 对存储空间预占用
hold on;box on;axis([0,pi*2,0,pi*2]); % 打开空的坐标轴
for d=1:M;
   y=d/M*pi*2;  % 设置 yn 的初值
   for c=1:M;
       x=c/M*pi*2;  % 设置 yn 的初值
       for n=1:N;
           yt=y+K*sin(x); % 按迭代式计算
           xt=x+yt;  % 按迭代式计算
           x=xt; % 更新 xn
           y=yt; % 更新 yn
           Zz(n)=mod(xt,2*pi)+mod(yt,2*pi)*i; % 以复数的形式记录数据
```

```
      end
      plot(Zz,'r.','markersize',1); % 以点的形式画图
   end
end
title(['{\itK}=',num2str(K)],'Fontsize',14,'Fontname','Times new roman');% 标注图题
```

执行上述程序所得图形如图 20.14 左图所示，当把参数 K 换为 1 时可以得到如图 20.14 右图所示图形（其中颜色设置为洋红色，其他参数未改动）。

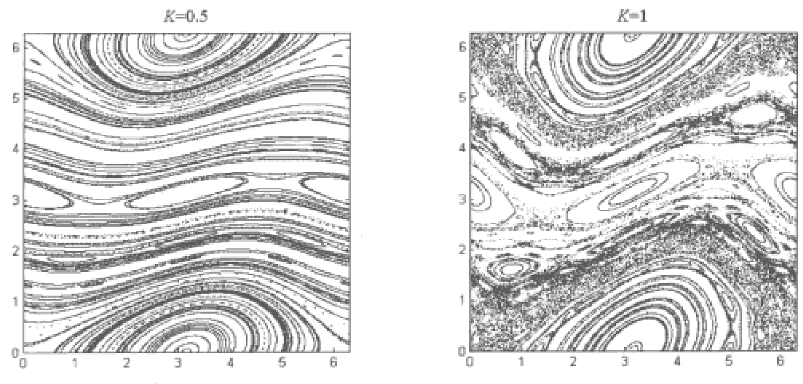


图 20.14 标准映射的相图

20.2 微分方程中的分岔和混沌行为

上一节介绍了离散形式的混沌行为，本节来介绍一些微分方程中表现出来的混沌行为。在本章第一节中已经说明了在很多非线性问题中可能出现混沌现象，而利用微分方程可以描述一些动力学或者相关过程的变化。本节举例说明根据微分方程绘制分岔图形的做法。

20.2.1 根据微分方程绘制分岔图形的做法——举例说明

首先以受驱单摆为例，对应方程可以写为：

$$\frac{d^2\theta}{dt^2}+\frac{1}{q}\frac{d\theta}{dt}+\sin\theta=F\cos\nu t \qquad (20\text{-}14)$$

其中 θ 是单摆的角位移，q 是品质因子，$1/q$ 表示阻尼因子，F 是驱动力的振幅，ν 是驱动力的频率。q，F 和 ν 是控制单摆的参数。这里考虑设置 $q=2$，$\nu=2/3$，而 F 在区间 $[0.96,1.52]$ 内取值。在求解微分方程时，初始条件为 $\theta(0)=0$，$\theta'(0)$ 在区间 $[0,2]$ 内取不同的值。而时间 t 计算的范围是 $[0,66]$。需要计算出 $\theta'(66)$ 和 F 的关系图。首先定义微分方程：

```
function dx=pendulum(t,x,flag,F);
dx=[x(2);F*cos(2*t/3)-0.5*x(2)-sin(x(1))];
```

调用函数 ode45 求解上面的微分方程，可以绘制分岔图，相应程序内容如下：

```
figure;hold on;box on;xlim([0.96,1.52]);% 生成空的坐标轴
N=80;          % 设置迭代次数
F1=ones(1,N); % 生成空的数组
```

```
for F=0.96:.001:1.52;
    for n=1:N;
        [t,x]=ode45('pendulum',[0,66],[0,n/N*2],[],F); % 求解微分方程
        T(n)=x(end,2);  % 提取关键点
    end
    F  % 实时显示 F 的取值
    plot(F1*F,T,'k.','markersize',1); % 绘图
    pause(0.01);                % 暂停一下
end
xlabel('\itF','Fontsize',16,'Fontname','Times new roman'); % X 轴标注
ylabel('{\it\theta} \prime','Fontsize',16,'Fontname','Times new
roman','Rotation',0); % Y 轴标注
set(gca,'Xtick',0.96:0.07:1.52); % 设置坐标轴刻度
```



上述程序执行起来比较耗时间。

程序结束后输出图形如图 20.15 所示。

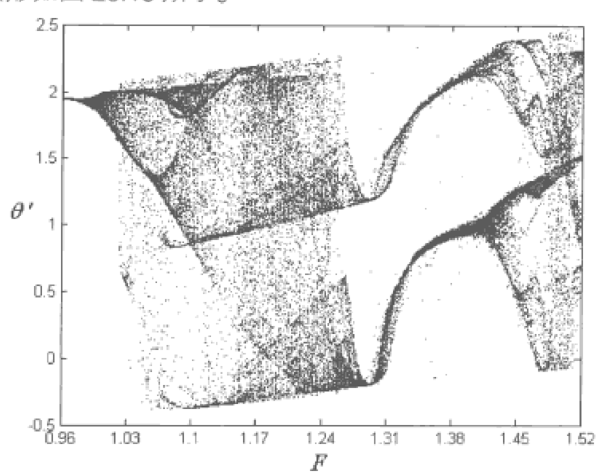


图 20.15 受驱单摆角速度关于驱动力的分岔图

下面再举一例说明微分方程中的分岔现象，微分方程组如下：

$$x''' = -Bx'' - x' - (|x| - 1) \tag{20-15}$$

其中参数 B 的取值范围是 $[0.57, 0.7]$，微分方程的初始条件为 $x = x' = x'' = 0$。

绘制这个分岔图的思路如下：逐个计算参数 B，在每次求解微分方程时，先使方程进入稳定状态，然后计算一个小的时间区间，如 $[0, 5]$，并求出区间内 x 的最大值，多次迭代计算最大值。将相应的最大值显示出来就可以得到相应的分岔图。如果所有最大值相等，则对应一条弦，否则就是分岔现象或者进入混沌状态。

根据上面的说明，可以写出分岔图的程序，相应的 MATLAB 程序内容如下：

```
hold on;box on;xlim([0.57,0.75]);ylim([1.3,2.7]);set(gca,'Xdir','reverse')% 生成
空的坐标轴
N=100; % 设置迭代次数
Tn=[];  % 初始化 Tn
options=odeset('RelTol',1e-9); % 设置微分方程求解选项
```

```
for B=0.75:-.0001:0.57; % 计算不同的 B 值
    x0=[0,0,0];         % 设置初值
    for n=1:200;        % 进行预迭代计算
        [t,x]=ode45('dfun1',[0,5],x0,options,B); % 预迭代解微分方程使之达到稳定状态
        x0=x(end,:);  % 更新初值
    end
    for n=1:N;
        [t,x]=ode45('dfun1',[0,5],x0,options,B); % 求解微分方程
        x0=x(end,:); % 更新初值
        xd=x(:,1);   % 提取求解结果
        [M,Ik]=max(xd); % 计算最大值
        if 1<Ik & Ik<length(xd); % 判断是否为最大值内部点
            Tn=[Tn;M]; % 记录最大值
        end
    end
    plot(B,Tn,'k.','markersize',1); % 绘图
    Tn=[]; % 清空
    pause(0.01);               % 暂停一下
end
xlabel('\itB','Fontsize',16,'Fontname','Times new roman');            % X 轴标注
ylabel('{\itx}','Fontsize',16,'Fontname','Times new roman','Rotation',0); % Y 轴
标注
```

与离散情况类似，先进行一定次数的预迭代求解微分方程，然后开始以时间区间[0, 5]解微分方程并找出最大值进行绘图，从而可以得到分岔图。执行上述程序输出图形如图 20.16 所示，可见在参数 B 取不同数值时该微分方程可以出现分岔和混沌行为。

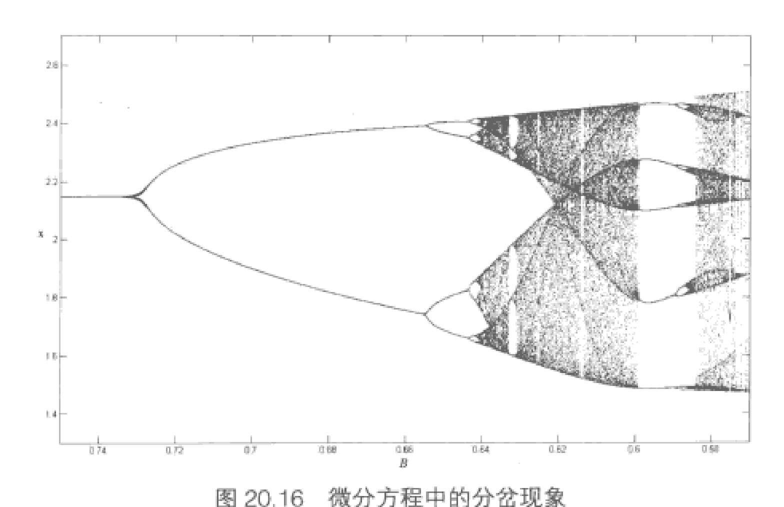


图 20.16 微分方程中的分岔现象

20.2.2 三元微分方程组中的分岔、混沌现象的模拟

下面给出一个三元微分方程组中的分岔、混沌现象的模拟，微分方程组定义如下：

$$\begin{cases} x'=y \\ y'=-2\zeta\omega y-\gamma\omega^2 x-(1-\gamma)\omega^2 z+B\sin pt \\ z'=Ay-\alpha|y|z-\beta y|z| \end{cases} \quad (20\text{-}16)$$

在模拟中，参数取值如下：$\varsigma=0.02$，$\omega=1$，$\gamma=0.5$，$p=1$，$A=1$，$\alpha=0.15$，

$\beta = -0.85$，而参数 B 的取值范围是 $[0,120]$。计算 x' 和 B 之间的关系图形。

因为这个微分方程组是在外部驱动力 $B\sin(pt)$ 的作用下进行的动力学行为，因此系统稳态运动的周期是 $2\pi/p=2\pi$。在求解这个微分方程的分岔图时，用函数 ode45 求解时时间计算范围取 $[0,2\pi]$。在计算点 x' 最大值时，先多次迭代求解微分方程，使之达到稳态运动，然后计算周期 2π 内的最大值。遍历一定范围内 B 的值，可以得出 x' 和 B 之间的分岔关系图形。

```
function dx=dfun2(t,x,flag,B);
xi=0.02;
gama=0.5;
omiga=1;
p=1;
alpha=0.15;
beta=-0.85;
A=1;
dx(1,1)=x(2);
dx(2,1)=-2*xi*omiga*x(2)-gama*omiga^2*x(1)-(1-gama)*omiga^2*x(3)+B*sin(p*t);
dx(3,1)=A*x(2)-alpha*abs(x(2))*x(3)-beta*x(2)*abs(x(3));
```

调用函数 ode45 求解微分方程，相应的实现程序如下：

```
hold on;box on;xlim([35.2,36.8]);%ylim([-20,10]);% 生成空的坐标轴
N=80;         % 设置迭代次数
Tn=zeros(1,N); % 初始化 Tn
options=odeset('RelTol',1e-9);    % 设置微分方程求解选项
for B=35.2:.002:36.8;            % 计算不同的 B 值
    x0=[0,0,0]; % 设置初值
    for n=1:60; % 进行预迭代计算
        [t,x]=ode45('dfun2',[0,pi*2],x0,options,B); % 预迭代解微分方程使之达到稳定状态
        x0=x(end,:); % 更新初值
    end
    for n=1:N;
        [t,x]=ode45('dfun2',[0,pi*2],x0,options,B); % 求解微分方程
        x0=x(end,:); % 更新初值
        xd=x(:,2);   % 提取求解结果
        Tn(n)=max(xd);
    end
    B
    plot(B,Tn,'k.','markersize',1); % 绘图
    pause(0.01);                 % 暂停一下
end
xlabel('\itB','Fontsize',16,'Fontname','Times new roman'); % X 轴标注
ylabel('{\itx}\prime','Fontsize',16,'Fontname','Times new roman','Rotation',0);
% Y 轴标注
```

首先迭代求解微分方程，然后开始解方程并记录数据进行绘图。这里每次调用函数 ode45 求解微分方程的时间区间是[0, pi*2]，同时记录的数据位置是区间内的最大值处。执行上述程序所得图形如图 20.17 所示，对于这个微分方程系统同样地出现分岔和混沌行为。

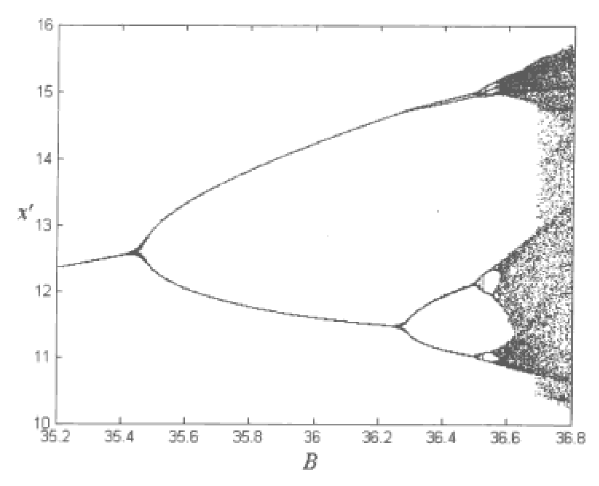


图 20.17　微分方程组对应的分岔图形

20.2.3　蔡氏混沌电路

蔡氏混沌电路是混沌研究中一个典型的电路模型，其最早由美籍华人蔡少棠教授提出。该方程的标准化形式为：

$$\begin{cases} x' = a[y - x - h(x)] \\ y' = x - y + z \\ z' = -by \end{cases} \tag{20-17}$$

其中 $h(x) = m_1 x + \frac{1}{2}(m_0 - m_1)(|x+1| - |x-1|)$ 在计算中参数取值为：$m_0 = -1.1$，$m_1 = -0.6$，$b = 12$，$a \in [7, 8.5]$。

首先定义微分方程，程序如下：

```
function dx=chua(t,x,flag,a,b);
m0=-1.1;
m1=-0.6;
hx=m1*x(1)+0.5*(m0-m1)*[abs(x(1)+1)-abs(x(1)-1)]; % 计算函数h(x)的值
dx=[a*(x(2)-x(1)-hx);x(1)-x(2)+x(3);-b*x(2)]; % 定义微分方程
```



这里把参数 a 和 b 作为方程的输入参数，在迭代计算中可以更改它们的数值。

调用函数 ode45 求解上面的微分方程，程序内容如下：

```
hold on;box on;xlim([7,8.5]);% 生成空的坐标轴
N=100;  % 设置迭代次数
Tn=[];  % 初始化 Tn
b=12;
options=odeset('RelTol',1e-9); % 设置微分方程求解选项
for a=7:0.002:8.5;  % 计算不同的 a 值
    x0=[0,0.3,0];  % 设置初值
```

```
    [t,x]=ode45('chua',[0,100],x0,options,a,b);
    x0=x(end,:);
    for n=1:N;
        [t,x]=ode45('chua',[0,2],x0,options,a,b); % 求解微分方程
        x0=x(end,:); % 更新初值
        xd=x(:,1);  % 提取求解结果
        [M,Ik]=max(xd); % 计算最大值
        if 1<Ik & Ik<length(xd); % 判断是否为最大值内部点
            Tn=[Tn;M]; % 记录最大值
        end
    end
    plot(a,Tn,'k.','markersize',1); % 绘图
    Tn=[];        % 清空
    pause(0.01);  % 暂停一下
end
xlabel('\ita','Fontsize',16,'Fontname','Times new roman');             % X 轴标注
ylabel('{\itx}','Fontsize',16,'Fontname','Times new roman','Rotation',0); % Y 轴
标注
set(gcf,'Color','w')
% set(gca,'Xtick',0.96:0.07:1.52); % 设置坐标轴刻度
axes('Position',[0.23,0.3,0.4,0.4]); % 建立一个小坐标轴，在其上绘制局部放大图
xlim([7.5,8]);hold on;box on; % 设置坐标轴属性
for a=7.5:0.0008:8; % 计算不同的 a 值
    x0=[0,0.3,0];  % 设置初值
    [t,x]=ode45('chua',[0,100],x0,options,a,b);
    x0=x(end,:);      % 获取迭代过程中的初值
    for n=1:N;
        [t,x]=ode45('chua',[0,2],x0,options,a,b); % 求解微分方程
        x0=x(end,:); % 更新初值
        xd=x(:,1);      % 提取求解结果
        [M,Ik]=max(xd); % 计算最大值
        if 1<Ik & Ik<length(xd); % 判断是否为最大值内部点
            Tn=[Tn;M];        % 记录最大值
        end
    end
    plot(a,Tn,'k.','markersize',1); % 绘图
    Tn=[];                    % 清空临时数据
    pause(0.01);               % 暂停一下
end
xlabel('\ita','Fontsize',16,'Fontname','Times new roman'); % X 轴标注
```

这里先进行时间区间为[0, 100]时的微分方程求解，使之进入稳定状态，然后调用 ode45 函数在范围较小区间内进行求解并记录区间内的最大值进行绘图。为了突出细节部分的情况，这里将参数 a 在[0.75, 8]范围的小范围图形进行放大并显示在一个小的坐标轴内。执行上述程序输出的图形如图 20.18 所示，从图中可以看出分岔和混沌现象随 a 变化的情况。

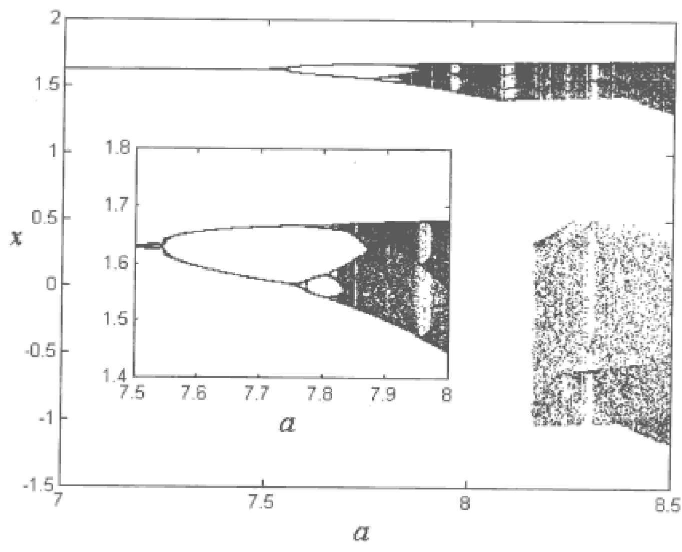


图 20.18 蔡氏电路中的分岔现象

20.3 混沌吸引子

本节介绍混沌吸引子方面的知识。其中相图和庞加莱截面是两个比较重要的概念，首先来介绍相图和庞加莱相关的知识。

20.3.1 相图

在介绍吸引子之前，先给出相图的定义。一般情况下，微分方程或者方程组可以表达为不显含时间 t 的形式：

$$\begin{cases} x_1' = f_1(x_1,\ldots,x_n,x_1',\ldots,x_n') \\ x_2' = f_2(x_1,\ldots,x_n,x_1',\ldots,x_n') \\ \ldots\ldots \\ x_n' = f_n(x_1,\ldots,x_n,x_1',\ldots,x_n') \end{cases} \tag{20-18}$$

通过求解微分方程组可以得到 $x_1(t)$，$x_2(t)$，…，$x_n(t)$ 的值，按时间顺序在 n 维空间内绘制相应的曲线，所得图形被称为相图。曲线上的点表示系统在相应时刻的状态，该图形上的曲线被称为相轨线。通过分析相轨线可以了解系统是否存在混沌现象。对于离散的映像，k 表示迭代次数，在上脚标的括号中表示：

$$\begin{cases} x_1^{(k+1)} = g_1(x_1^{(k)},\ldots,x_n^{(k)}) \\ x_2^{(k+1)} = g_2(x_1^{(k)},\ldots,x_n^{(k)}) \\ \ldots\ldots \\ x_n^{(k+1)} = g_n(x_1^{(k)},\ldots,x_n^{(k)}) \end{cases} \tag{20-19}$$

可以根据迭代的数值序列 x_1，x_2，…，x_n 绘图，所得图形即为该映射的相图。

下面举例说明埃农映射相图的绘制，相关程序如下，绘制的图形如图 20.19 所示。

```
a=1.127;  % 对参数 a 赋值
b=0.3;     % 对参数 b 赋值
N=10000; % 设置迭代次数
xn=zeros(1,N); % 预置空间
yn=zeros(1,N); % 预置空间
x=0;y=0;       % 设置预迭代初值
for k=1:800;
    xt=1-a*x^2+y;  % 进行预迭代
    yt=b*x;        % 进行预迭代
    x=xt;  % 更新迭代值
    y=yt;  % 更新迭代值
end
xn(1)=x;  % 记录迭代初值
yn(1)=y;  % 记录迭代初值
for k=1:N-1;
    xn(k+1)=1-a*xn(k)^2+yn(k); %记录迭代数据
    yn(k+1)=b*xn(k);           %记录迭代数据
end
plot(xn,yn,'k.','Markersize',1); % 绘图
```

关于微分方程组对应的相图的绘制在下面两个小节中介绍。

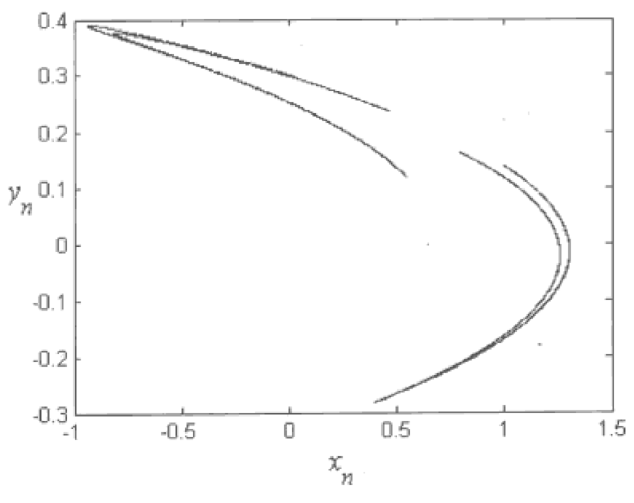


图 20.19 埃农映射相图

研究相空间中的复杂轨线，庞加莱发展了一种截面方法。该方法是在相空间内取某一坐标为常数的截面，通过研究相轨线和该截面的交点来分析系统演化的过程。如果相空间是 n 维的，那么可以取出 n-1 维空间的相平面，该平面就是庞加莱截面。

这里以单摆运动方程为例说明庞加莱截面的绘制，单摆方程可以写为：

$$\theta'' + \gamma\theta' + \omega^2 \sin\theta = F\cos\nu t \tag{20-20}$$

引入一个相位函数量 φ ，上面的方程可以转化为下面不显含时间的系统：

$$\begin{cases}\theta'=\omega\\ \omega'=F\cos\varphi-\gamma\omega-\omega^2\sin\theta\\ \varphi'=\nu\end{cases} \tag{20-21}$$

在相空间(θ,ω,φ)中，取$\varphi=0$作为庞加莱截面。值得注意的是变量φ以一个周期函数的形式$\cos\varphi$（其周期为2π），因此相曲线每通过$\varphi=2n\pi$就可以在庞加莱截面$\varphi=0$留下一点，这样取时间区间为$[0,2\pi]$多次迭代求解方程组，每次求解时间终端处的值就可以得到相曲线与庞加莱截面的交点。

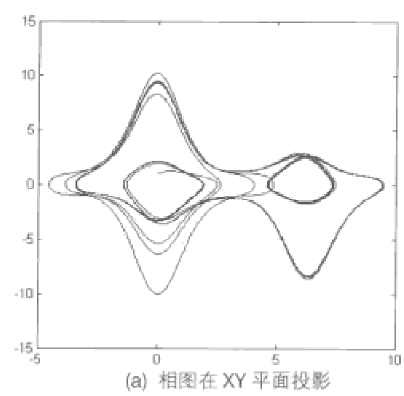
首先定义单摆微分方程对应的函数文件 pppp.m，即：

```
function dx=pppp(t,x,flag,F,gama,v);
dx=[x(2);F*cos(x(3))-gama*x(2)-x(2)^2*sin(x(1));v];  % 定义微分方程组
```

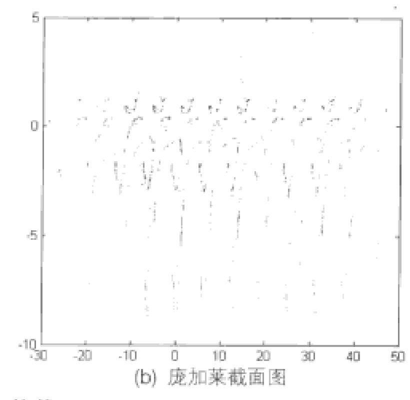
根据上面的思路可以得到求解庞加莱截面的程序。

```
F=1.49;gama=0.5;v=2/3;        % 设置单摆方程组的参数
options=odeset('RelTol',1e-9);  % 设置微分方程求解选项
[t,x]=ode45('pppp',[0,200],[0,1,0],options,F,gama,v);  % 求解单摆方程
subplot(121);plot(x(:,1),x(:,2));
xlabel('(b)','Fontsize',16,'Fontname','Times new roman');% X 轴标注
x0=x(end,:);  % 从前面求解结果中获取初值
N=4000;   % 设置迭代次数
xk=ones(1,N)*x0(1);  % 预置空间
yk=ones(1,N)*x0(2);  % 预置空间
for k=1:N-1;
    [t,x]=ode45('pppp',[0,pi*2],x0,[],F,gama,v);  % 求解方程
    x0=x(end,:);   % 获取解的终端值
    xk(k+1)=x0(1);  % 提取与庞加莱截面的交点坐标
    yk(k+1)=x0(2);  % 提取与庞加莱截面的交点坐标
end
subplot(122);plot(xk,yk,'r.','markersize',1);  % 绘制庞加莱截面上的交点图
xlabel('(b)','Fontsize',16,'Fontname','Times new roman');  % X 轴标注
```

因为 phi 的周期等于 2*pi，所以在求解微分方程时以区间[0, pi*2]作为输入，通过提取区间内解最后一个数据就是相应在庞加莱截面的投影。多次计算后可以得到庞加莱截面上的图案。执行上述程序，输出图形如图 20.20 所示，其中图(a)是角度和角加速度关系曲线，散乱的图(b)是庞加莱截面上的图案。



(a) 相图在 XY 平面投影



(b) 庞加莱截面图

图 20.20 庞加莱截面图



本例中利用了分量φ的周期性，使得计算过程变得比较简单。如果分量不存在周期性，可以考虑利用插值的方法获得与庞加莱截面的交点坐标。

20.3.2 Lorenz 吸引子

美国数学家洛伦兹（E. N. Lorenz）把大气对流和贝纳德液体对流联系起来，利用流体力学中的纳维叶-斯托克斯（Navier-Stokes）方程、热传导方程和连续性方程，推导出著名的洛伦兹（Lorenz）方程组，其数学定义如下：

$$\begin{cases} x' = -\sigma(x-y) \\ y' = rx - y - xz \\ z' = -bz + xy \end{cases} \tag{20-22}$$

其中σ，r和b是系数，它们都是正数。

令$x' = y' = z' = 0$，可以得到洛伦兹方程的平衡点，即：

$$\begin{cases} -\sigma(x-y) = 0 \\ rx - y - xz = 0 \\ -bz + xy = 0 \end{cases} \tag{20-23}$$

求解上面的方程可以得到三组解，即：

$$\begin{cases} x = 0 \\ y = 0 \\ z = 0 \end{cases},\quad \begin{cases} x = \sqrt{b(r-1)} \\ y = \sqrt{b(r-1)} \\ z = r-1 \end{cases},\quad \begin{cases} x = -\sqrt{b(r-1)} \\ y = -\sqrt{b(r-1)} \\ z = r-1 \end{cases} \tag{20-24}$$

这样洛伦兹方程组平衡点是：$(0,0,0)$，$\left(\sqrt{b(r-1)},\sqrt{b(r-1)},r-1\right)$和$\left(-\sqrt{b(r-1)},-\sqrt{b(r-1)},r-1\right)$。

首先定义洛伦兹微分方程组，程序如下：

```
function dx=lorenzf(t,x,flag,sigma,r,b);
dx=[-sigma*(x(1)-x(2));r*x(1)-x(2)-x(1)*x(3);-b*x(3)+x(1)*x(2)]; % 定义微分方程组
```



把上面两行语句存储为 lorenzf.m 文件。在 MATLAB 中提供了函数 lorenz 来演示洛伦兹吸引子的绘制，读者在命令窗中输入 lorenz 即可运行这个演示。

在计算中参数取值为$\sigma = 9$，$r = 27$，$b = 3$，初始条件为$x_{t=0} = 1$，$y_{t=0} = z_{t=0} = 0$。利用下面的程序可以求解洛伦兹微分方程组并绘图。

```
x0=[1,0,0]; % 定义初值
sigma=9; % 对参数σ赋值
r=27; % 对参数r赋值
b=3; % 对参数b赋值
[t,x]=ode45('lorenzf',[0,60],x0,[],sigma,r,b); % 求解洛伦兹微分方程组
```

```
subplot(221);plot(t,x(:,1));L(1)=xlabel('\itt');L(2)=ylabel('\itx'); % 绘制t ~ x 曲线
subplot(222);plot(t,x(:,2));L(3)=xlabel('\itt');L(4)=ylabel('\ity'); % 绘制t ~ y 曲线
subplot(223);plot(t,x(:,3));L(5)=xlabel('\itt');L(6)=ylabel('\itz'); % 绘制t ~ z 曲线
t(1:1500)=[];x(1:1500,:)=[]; % 删去前面一段数据
subplot(224);plot3(x(:,1),x(:,2),x(:,3));box on; % 绘制三维相轨线图
L(7)=xlabel('{\itx}');L(8)=ylabel('{\ity}');L(9)=zlabel('{\itz}');% X轴、Y轴和Z 轴标注
set(L,'Fontsize',12,'Fontname','Times New Roman');% 设置字体属性
```

输出图形如图 20.21 所示。

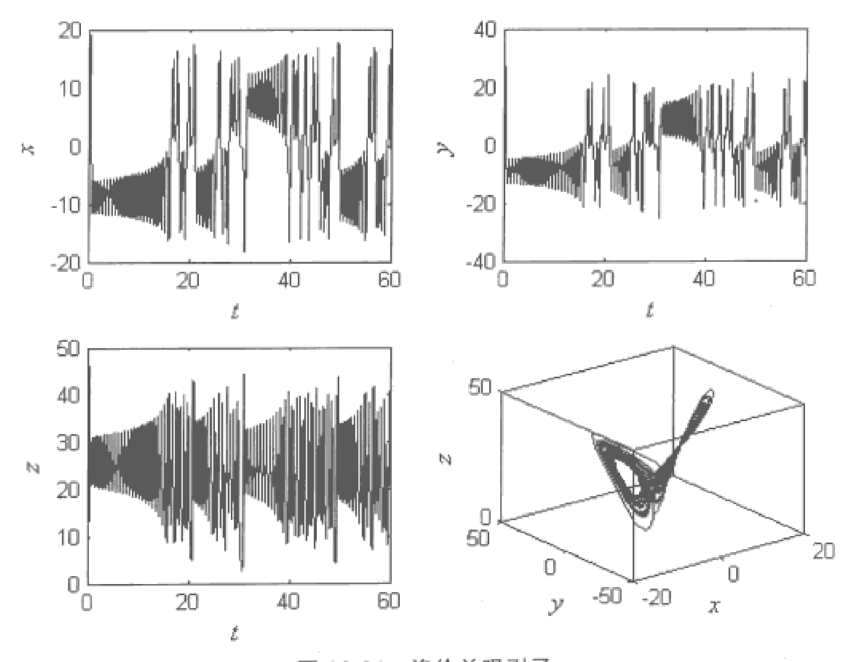


图 20.21 洛伦兹吸引子

可见方程的解围绕空间中两点不断地在两个区域往复运动，读者可以使用函数 comet3 来动态绘制洛伦兹吸引子以观察其状态变化过程。

20.3.3 Rossler 吸引子

在 1976 年，罗斯勒（O.E. Rössler）在洛伦兹方程组的基础上设计了一个新的吸引子方程，即罗斯勒方程组，其数学描述如下：

$$\begin{cases} x' = -(y+z) \\ y' = x + ay \\ z' = b + (x-c)z \end{cases} \tag{20-25}$$

下面考虑求解罗斯勒方程组，首先定义函数文件 rossler.m 描述这个微分方程组。

```
function dx=rossler(t,x,flag,a,b,c);
dx=[-x(2)-x(3);x(1)+a*x(2);b+(x(1)-c)*x(3)];   % 定义微分方程
```

这里方程组的参数取值为：a=b=1，c=6，7，8，初值条件为 $x_{t=0}=1$，$y_{t=0}=z_{t=0}=0$。然后利用下面的程序就可以绘制罗斯勒吸引子。

```
x0=[1,0,0]; % 定义初值
a=0.1; % 对参数 a 赋值
b=0.1; % 对参数 b 赋值
[t,x]=ode45('rossler',[0,100],x0,[],a,b,6);x(t<50,:)=[]; % 求解洛伦兹微分方程组，并把解中 t<50 的部分数据删去
subplot(131);plot3(x(:,1),x(:,2),x(:,3));ti(1)=title('{\itc}=6'); % 绘制曲线
[t,x]=ode45('rossler',[0,100],x0,[],a,b,7);x(t<50,:)=[]; % 求解洛伦兹微分方程组，并把解中 t<50 的部分数据删去
subplot(132);plot3(x(:,1),x(:,2),x(:,3));ti(2)=title('{\itc}=7'); % 绘制曲线
[t,x]=ode45('rossler',[0,100],x0,[],a,b,8);x(t<50,:)=[]; % 求解洛伦兹微分方程组，并把解中 t<50 的部分数据删去
subplot(133);plot3(x(:,1),x(:,2),x(:,3));ti(3)=title('{\itc}=8'); % 绘制曲线
set(ti,'Fontsize',12,'Fontname','Times New Roman');% 设置字体属性
```

执行上述程序输出图形如图 20.22 所示。

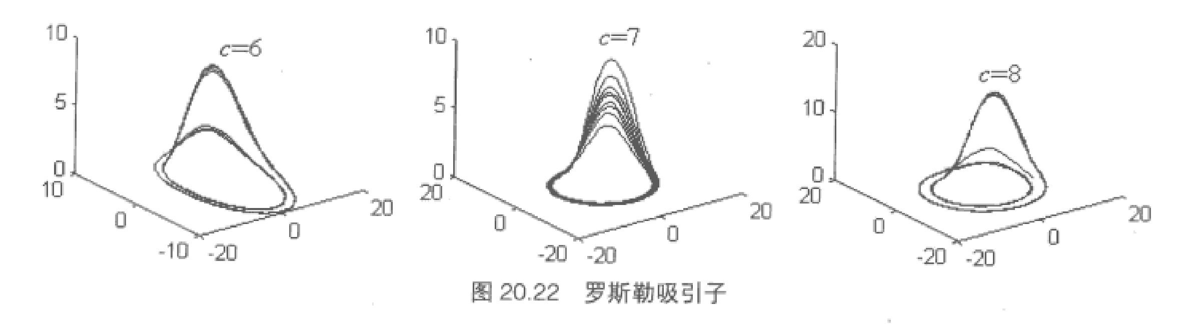


图 20.22 罗斯勒吸引子

20.4 Lyapunov 指数

李雅普诺夫（Lyapunov）指数是描述混沌现象的一个重要指标。一个映射 $x=f(x,a)$ 的李雅普诺夫指数 λ 的计算公式为：

$$\lambda=\lim_{n\to\infty}\frac{1}{n}\sum_{k=0}^{n-1}\ln\left|\frac{\mathrm{d}f(x_k,a)}{\mathrm{d}x}\right| \tag{20-26}$$

当 λ 是负数时，映射最后收敛于某一点；当 λ 等于 0 时，映射进行周期运动；当 λ 是正数时，映射处于混沌状态。对于在计算机上进行李雅普诺夫指数 λ 的计算，只考虑计算 n 为较大数即可，而不必计算无穷多个点的导数。本节将给出映射和微分方程组的李雅普诺夫指数 λ 的计算例子。

这里以罗杰斯蒂映射为例，该映射关系的导数表达式为：

$$\frac{\mathrm{d}f(x,a)}{\mathrm{d}x}=a-2ax \tag{20-27}$$

在计算中参数 n 的取值为 600，迭代过程中的初值为 0.1。这里参数 a 的取值范围是[3.4, 4]。为了便于比较李雅普诺夫指数与系统状态的关系，此处同时把映射图形给出。

具体计算程序如下：

```
n=600;              % 设置计算李雅普诺夫过程中迭代的总次数
```

```
xn=zeros(1,n);  % 预置全 0 数组占据空间
aa=3.4:0.001:4; % 离散化参数 a
N=1;          % 计数指标
subplot(211);hold on;box on;xlim([min(aa),max(aa)]);     % 生成空的坐标
XL(1)=ylabel('\itx');
for a=aa;
    x=0.1;           % 预置初值
    for q=1:80;
        x=a*x*(1-x);  % 预迭代
    end
    s=0;            % 把累加参数设置为 0
    for q=1:n;
        xn(q)=x;    % 记录映射的绘图数据
        df=a-2*a*x; % 计算导数值
        s=s+log(abs(df));  % 累加绝对值的对数
        x=a*x*(1-x);       % 迭代计算下一个值
    end
    L(N)=s/n; % 计算李雅普诺夫指数
    N=N+1;  % 计数器累加
    plot(a,xn(1:100),'k.','markersize',1); % 绘制映射图
    a,pause(0.01)
end
subplot(212);plot(aa,L);    % 在新建坐标轴上绘制李雅普诺夫指数曲线
XL(2)=xlabel('\ita');XL(3)=ylabel('\it\lambda');
set(XL,'Fontsize',14,'Fontname','Times new Roman');
hold on;box on;plot(xlim,[0,0],'r--');    % 绘制 0 刻度线
xlim([min(aa),max(aa)]); % 设置坐标轴范围
```

执行上述程序输出图形如图 20.23 所示。本程序运行比较耗时，需要耐心等待运行结果。

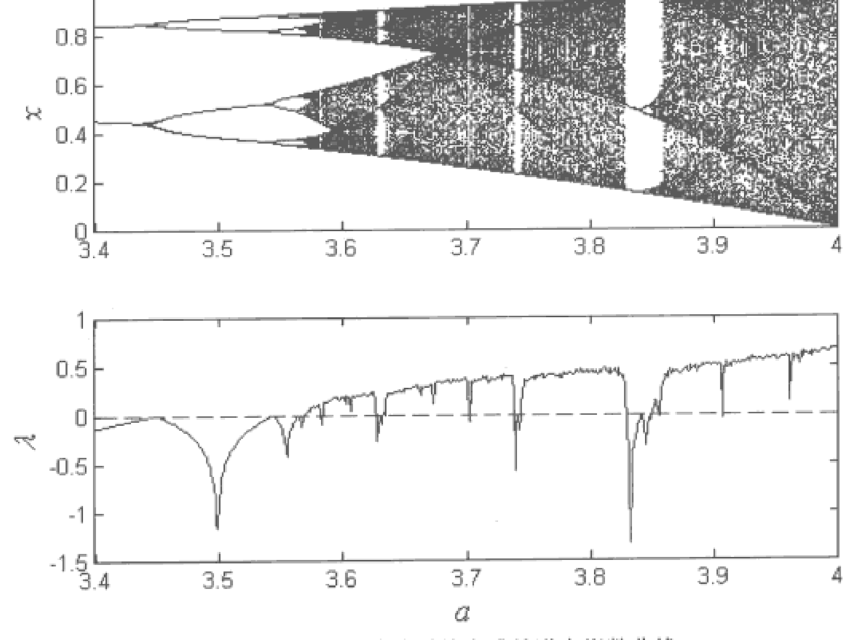


图 20.23　罗杰斯蒂映射的李雅普诺夫指数曲线

对于微分方程组的李雅普诺夫指数的计算可以利用 Wolf 等提出的基于 Gram_Schmidt 正交化的求解方法。该算法实现程序的下载网址为 http://www.mathworks.com/matlabcentral/fileexchange/4628。读者可以下载到一个压缩包，其中含有 3 个文件，即 lorenz_ext.m，lyapunov.m 和 run_lyap.m，前两个文件是函数文件，最后一个文件是脚本文件。下面分别介绍这 3 个文件的内容。

（1）函数文件 lorenz_ext 是定义微分方程组以及对应的雅可比（Jacobi）矩阵，该函数的调用格式为：

```
f=lorenz_ext(t,X);
```

参数说明： f 是一个 $(n+1)\times n$ 矩阵，其中 n 是时间函数的个数，第一行的 n 个数表示离散微分值，后面 $n\times n$ 是微分方程对应的雅可比矩阵，比如微分方程可以表示为：

$$x_k' = f_k\left(x_1,\ldots,x_n,x_1',\ldots,x_n'\right) \tag{20-28}$$

其中 $1\le k\le n$，雅可比矩阵 $J_{p,q}$ 的元素是：

$$J_{p,q} = \frac{\partial f_p}{\partial x_q} \tag{20-29}$$

其中 p 和 q 是正整数，且 $1\le p,q\le n$。

t 是时间值，x 是由待求函数的值组成的向量。这个程序包是针对洛伦兹方程的，如果想计算其他微分方程组，可以相应地替换 lorenz_ext 文件内容得到相应函数的表达式。

（2）函数文件 lyapunov.m 是用来计算李雅普诺夫指数的程序，该函数的调用格式为：

```
[Texp,Lexp]=lyapunov(n,rhs_ext_fcn,fcn_integrator,tstart,stept,tend,ystart,iou
tp);
```

参数说明： Texp 是时间的数值。Lexp 是李雅普诺夫指数的数值。n 表示微分方程的个数。rhs_ext_fcn 是前面定义微分方程组的函数句柄名。fcn_integrator 是所调用求微分方程的 MATLAB 函数名。tstart 是时间的初始值。stept 是进行 Gram_Schmidt 正交化的迭代次数。tend 是时间的终止数值。ystart 是求解微分方程的初值。ioutp 表示每隔 ioutp 个点输出相应的时间和李雅普诺夫指数数值。

（3）run_lyap.m 程序是调用前面两个函数计算李雅普诺夫指数的程序。

20.5 小结

本章主要介绍混沌现象的模拟。首先介绍了典型映射的分岔图的绘制方法，以及根据一些映射绘制分岔图的例子。然后介绍了微分方程中的混沌现象，通过变化微分方程中的参数绘制相应的混沌图形。混沌吸引子可以很好地刻画混沌性质，本章介绍了混沌相图、混沌吸引子的绘制方法以及庞加莱截面的画法。最后介绍了李雅普诺夫指数，并给出了映射和微分方程系统的李雅普诺夫指数的计算方法。

◆ ◆ ◆

第21章 分形图形

本章包括

- ◆ **基本分形图** 介绍康托集及其派生图形、Julia 集、Mandelbrot 集和 Koch 曲线等分形图的绘制。
- ◆ **迭代函数系统** 介绍利用迭代函数系统绘制不同类型的分形图形。
- ◆ **递归算法** 介绍利用递归算法绘制分形图形的方法。
- ◆ **分维的计算** 给出分形图形维数的计算方法。

分形的英文是 Fractal，它是 B.Mandelbrot 根据拉丁语的词根设计的一个单词。最初分形的研究是关于一些特殊图形的绘制。后来人们发现很多学科中的图案与分形有关，从而计算图案的分形维数（称为分维），不同的分维数对应着不同类型的意义。目前分形已经应用于很多领域，如数学、物理、材料科学、生物学、地理学和计算机科学等。本章主要介绍基本分形图的绘制方法以及分维的计算等知识。

21.1 基本分形图

了解分形应该从分形图的绘制开始。本节介绍几种经典分形的绘制方法，从中可以了解到分形图结构单元的内在联系。

21.1.1 康托集

康托（Cantor）集的定义是每次截去某行线段中间的 1/3 部分，这样就得到很多小线段。每次得到的线段长度是上面一行的 1/3。绘制的算法是计算各个线段的三等分点的数值就是新增点的坐标，即：

$$z_{n,p}=\frac{z_{n-1,q}}{3}+\frac{2z_{n-1,q+1}}{3}\text{，}\quad z_{n,p+1}=\frac{2z_{n-1,q}}{3}+\frac{z_{n-1,q+1}}{3} \tag{21-1}$$

其中，$z_{n,p}$，$z_{n,p+1}$是分割后的等分点坐标，$z_{n-1,q}$，$z_{n-1,q+1}$是分割点的线段端点坐标。

通过上面的介绍可以得到康托集的绘制程序，具体内容如下：

```
N=6;    % 分割次数
axis([0,8,-1,N+1]);hold on;box on;    % 生成空的坐标轴
Z=[1+N*i,7+N*i];  % 设置初始时坐标的数据，这里用复数表示坐标点
plot(Z,'r');  % 绘图
text(7.2,N,'{\itC}_0','Fontsize',14,'Fontname','Times New Roman');  % 标注
text(0.4,N,'{\itC}_0','Fontsize',14,'Fontname','Times New Roman');  % 标注
for k=1:N;
    Z=Z-i; % 下移一段距离
    for q=1:2:2^k;
        Zt(2*q-1)=Z(q); % 记录分割点端点坐标
```

```
        Zt(2*q)=2*Z(q)/3+Z(q+1)/3;    % 计算内部三等分点
        Zt(2*q+1)=Z(q)/3+Z(q+1)*2/3;  % 计算内部三等分点
        Zt(2*q+2)=Z(q+1);         % 记录分割点端点坐标
        L1=[Zt(2*q-1),Zt(2*q)];    % 计算等分线段的端点坐标
        L2=[Zt(2*q+1),Zt(2*q+2)]; % 计算等分线段的端点坐标
        plot(real(L1),imag(L1),'r',real(L2),imag(L2),'r'); % 画小线段
        text(7.2,N-k,['{\itC}_',num2str(k)],'Fontsize',14,'Fontname','Times New
Roman'); % 标注
        text(0.4,N-k,['{\itC}_',num2str(k)],'Fontsize',14,'Fontname','Times New
Roman'); % 标注
    end
    Z=Zt;    % 更新线段端点坐标
End
```

上述程序的整体思路是先初始化得到一条线段，以后每步计算中先得出每段线段的三等分点，然后进行连线。其中 L1 和 L2 分别是线段三等分后左侧线段和右侧线段的端点。同时每步计算中要把线段的高度减 1。执行上述程序输出图形如图 21.1 所示，可见线段的长度随着步数逐渐变小，缩小的比例为 1/3。

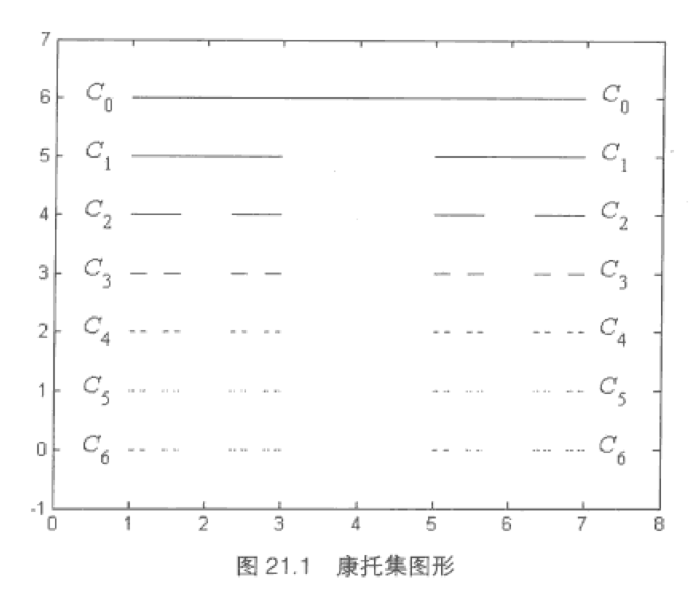


图 21.1　康托集图形

下面给出递归调用的程序实现。

```
function [za,zb]=cantor(za,zb);
if nargin==0;
   za=50+110i;    % 参数默认时左端点的坐标值
   zb=750+110*i; % 参数默认时右端点的坐标值
end
if real(za)>real(zb);
   error('zb 实部必须大于 za 的实部');
end
c=8;  % cantor 集中线段的最小长度，控制程序执行的深度
d=50; % 上下两层线段之间的距离
hold on;
if abs(real(za)-real(zb))>c;
   plot(real([za,zb]),imag([za,zb]),'r','linewidth',2); % 绘制线段
```

```
    zc=za*2/3+zb/3; % 计算三等分点坐标
    zc=zc-d*i; % 将线段右端点下移一段距离
    za=za-d*i; % 将线段左端点下移一段距离
    [za,zc]=cantor(za,zc); % 递归调用
    za=za+d*i;      % 将 za 点恢复到原来位置
    zd=za/3+zb*2/3; % 计算三等分点坐标
    zb=zb-d*i;  % 将线段左端点下移一段距离
    zd=zd-d*i;  % 将线段右端点下移一段距离
    [zd,zb]=cantor(zd,zb); % 递归调用
End
```

在每步递归中，把线段的两个三等分点与最近的端点进行组合而进入下一步的等分过程。根据这个思路可以设计出递归结构，同时在递归过程中需要考虑每步降低线段的高度。通过下面的语句调用上面的函数 Cantor，可以得到相应的康托集。

```
[za,zb]=Cantor(50+110i,750+110i); % 绘制康托集
ylim([-110,120]); box on;          % 设置 Y 轴范围
```

执行上述程序输出图形如图 21.2 所示，除了相邻层线段的高度不同外，图 21.2 的结构与图 21.1 一样。

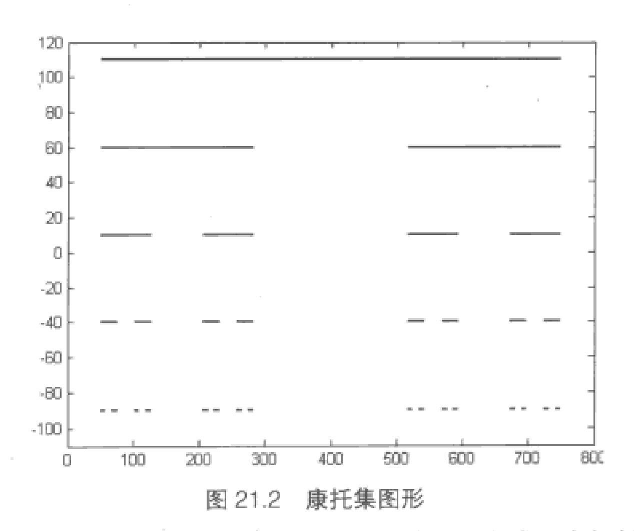


图 21.2　康托集图形

此外，光盘的\Ch21 文件夹下的 Cantor_general.m 文件可以生成一个倾斜的康托集。Cantor3.m 也是一个利用递归算法绘制康托集的程序。

下面来介绍二维康托集的绘制。在一个白色的方形区域，把区域等分为 3×3 的 9 个分块，涂黑中心的一个分块。按这个规则进行下去可以得到二维康托集。一个递归函数文件的内容如下：

```
function [zc,L]=Cantor_2d(zc,L);
if nargin==0;
   zc=0.5+0.5i; % 设置中心坐标值的默认值
   L=1/3;       % 设置正方形半边长的默认值
end
L0=0.02; % 最小半边长的界限
hold on;
[px,py]=meshgrid([-1,0,1]); % 生成相对坐标点
zm=[px+py*i]*L;             % 生成复数位移坐标
if L>L0;
```

```
    rectangle('position',[real(zc)-L/2,imag(zc)-L/2,L,L],'FaceColor','k'); % 填充区域
    zct=zc+zm(1);[zc,L]=Cantor_2d(zct,L/3);zc=zc-zm(1);L=L*3;  % 递归调用
    zct=zc+zm(2);[zc,L]=Cantor_2d(zct,L/3);zc=zc-zm(2);L=L*3;  % 递归调用
    zct=zc+zm(3);[zc,L]=Cantor_2d(zct,L/3);zc=zc-zm(3);L=L*3;  % 递归调用
    zct=zc+zm(4);[zc,L]=Cantor_2d(zct,L/3);zc=zc-zm(4);L=L*3;  % 递归调用
    zct=zc+zm(6);[zc,L]=Cantor_2d(zct,L/3);zc=zc-zm(6);L=L*3;  % 递归调用
    zct=zc+zm(7);[zc,L]=Cantor_2d(zct,L/3);zc=zc-zm(7);L=L*3;  % 递归调用
    zct=zc+zm(8);[zc,L]=Cantor_2d(zct,L/3);zc=zc-zm(8);L=L*3;  % 递归调用
    zct=zc+zm(9);[zc,L]=Cantor_2d(zct,L/3);zc=zc-zm(9);L=L*3;  % 递归调用
end
```



这里利用函数 rectangle 来填充方形区域内部颜色为黑色，读者还可以使用 fill 函数来实现区域的填充目的。

利用下面一段程序调用上面的 Cantor_2d 绘制二维康托集。

```
rectangle('position',[0,0,1,1],'LineStyle',':','EdgeColor','r'); % 画出考虑区域的边框
[zc,L]=Cantor_2d(0.5+0.5i,1/3); % 绘制二维康托集
axis(gca,'square','off');        % 设置坐标轴属性
set(gcf,'Color','w');           % 设置图形窗口为白色
```

首先利用函数 rectangle 生成正方形区域，边框利用虚线表示，再调用 Cantor_2d 绘制二维康托集图案，其中 Cantor_2d 的输入参数 zc 是中心点对应的复数，L 表示缩小比例。执行上述程序输出图形如图 21.3 所示，从中可以看出二维康托集具有分形图形的自相似性。

如果在3×3分块中填充如图 21.4 所示标记为“2”、“4”、“5”、“6”、“8”的区域为黑色，那么所得的分形图案可以利用下面的递归函数来绘制。

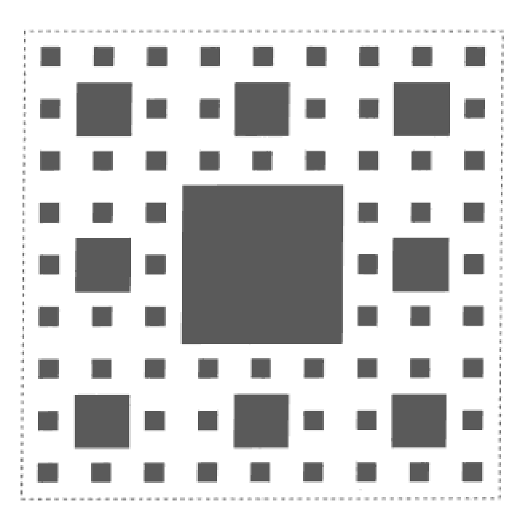


图 21.3　二维康托集的图案

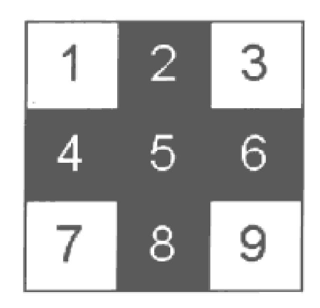


图 21.4　填充方案

```
function Cantor_Square_Fractal(n);
if nargin==0;
   n=4;  % 设置默认迭代次数
end
```

```
x=0;y=0; % 设置中心位置
hold on;
Cantor_Square(n,x,y,1); % 调用 Cantor_Square 函数绘分形图
axis square; % 设置坐标轴属性
function Cantor_Square(n,x,y,r);
HW=[-0.5,-0.5,1,1]; % 设置相对位移坐标
rr=r/3; % 设置单元网格的宽度
if n==0;
   rectangle('position',HW*r+[x,y,0,0],'FaceColor','k'); % 填充区域
else
   Cantor_Square(n-1,x,y,r/3);     % 递归调用
   Cantor_Square(n-1,x+rr,y,r/3);  % 递归调用
   Cantor_Square(n-1,x,y+rr,r/3);  % 递归调用
   Cantor_Square(n-1,x-rr,y,r/3);  % 递归调用
   Cantor_Square(n-1,x,y-rr,r/3);  % 递归调用
End
```

利用下面的程序调用函数 Cantor_Square_Fractal 以绘制相应的分形图：

```
subplot(131);box on;Cantor_Square_Fractal(2); % 绘制分形图
xlabel('n=2','HorizontalAlignment','center','Fontsize',14); % X 轴标注
subplot(132);box on;Cantor_Square_Fractal(3); % 绘制分形图
xlabel('n=3','HorizontalAlignment','center','Fontsize',14); % X 轴标注
subplot(133);box on;Cantor_Square_Fractal(4); % 绘制分形图
xlabel('n=4','HorizontalAlignment','center','Fontsize',14); % X 轴标注
```

Cantor_Square_Fractal 的设计与前面的 Cantor_2d 相似。先选定相应的区域利用函数 rectangle 把区域涂黑，然后调用函数 Cantor_Square_Fractal 作不同递归深度的图案。执行上述程序所得图形如图 21.5 所示，可见随着递归深度 n 的增加图案逐渐细致。

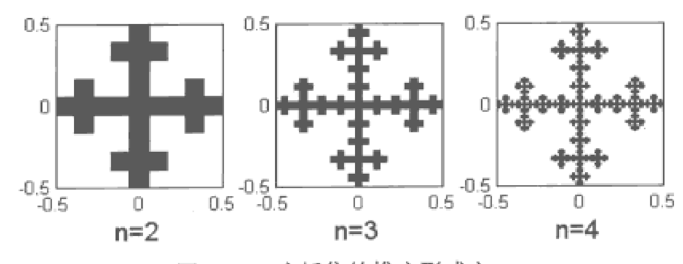


图 21.5　康托集的推广形式之一

21.1.2　Julia 集

Julia 集对应着下述映射关系：

$$z = z^2 + c \qquad (21\text{-}2)$$

其中 z 是复平面上的点，c 是一个复常数。Julia 图案的绘制过程可以总结为如下步骤。

step 1　选定一个区域并计算等间隔采样点。

step 2　把所有采样点带入公式（21-2）中，进行多次迭代。

step 3　当某个复数 z 的模值大于 M（预定义的一个大数，此时认为复数 z 的模值已经接近于"无穷"）时，记录相应的迭代次数 N。将各点对应 N 值，按大小关系着色即可得到相应的 Julia 图。下面给出绘制 Julia 集的函数文件 Julia.m 的内容。

```
function Julia(x,y,L,c);
if nargin==3;
   c=0.302-0.577i; % 设置参数 c 的默认值
end
Nmax=80; % 设置最大迭代次数
M=100;   % 设置一个大数，表示无穷大
[x1,y1]=meshgrid(linspace(x-L,x+L,512),linspace(y-L,y+L,512)); % 计算采样点坐标值
N=ones(size(x1))*Nmax; % 设置 N 的初值
z=x1+y1*i;             % 生成复数坐标点
for k=1:Nmax;
   z=z.^2+c;      % 迭代计算新的 z 值
   N(abs(z)>M)=k; % 找出模值超过 M 的点
end
image(linspace(x-L,x+L,512),linspace(y-L,y+L,512),N); % 根据 N 的数值绘图
title(['{\itc}=',num2str(c)]);
```

上面的函数 Julia 的调用格式为：

```
Julia(x,y,L);
Julia(x,y,L,c);
```

参数说明： x 和 y 是计算区域中心的横纵坐标。L 表示正方形区域的半边长。c 是 Julia 集的复常数取值，其默认值为 0.302-0.577i。当 x，y，L 和 c 取不同数值时，可以得到不同的 Julia 集的图案。

调用下面的程序可以得到如图 21.6 所示的图形。

```
subplot(221);Julia(0,0,2);axis square;xlabel('(a)');          % 绘制 Julia 集的图案
subplot(222);Julia(0,0.5,0.5);axis square;xlabel('(b)');      % 绘制 Julia 集的图案
subplot(223);Julia(-0.1,0.5,0.1);axis square;xlabel('(c)');   % 绘制 Julia 集的图案
subplot(224);Julia(-0.1,0.5,0.05);axis square;xlabel('(d)');  % 绘制 Julia 集的图案
```

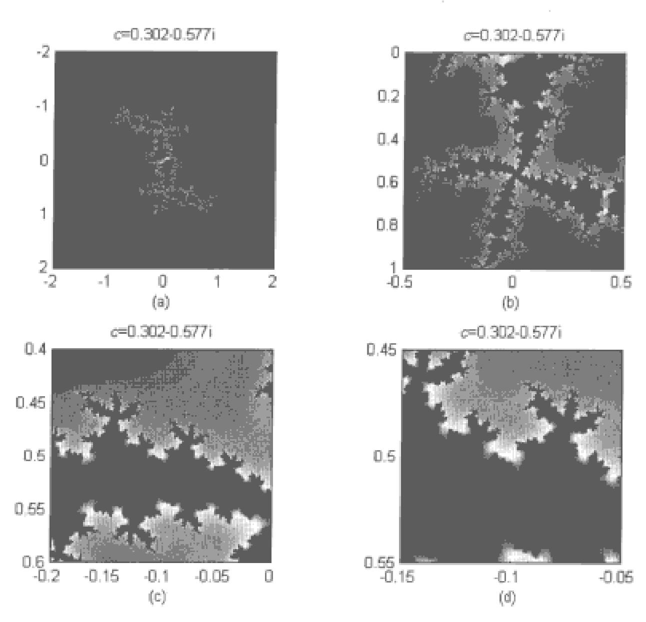


图 21.6　Julia 集的图案

下面给出一个制作 Julia 集的用户界面窗口的程序。利用这个程序，可以方便地改变前面 Julia.m

文件中的参数 x，y，L 和 c，并得到相应的分形图。首先应该把 Julia.m 文件进行修改，即：

```
function Julia(x,y,L,c);
if nargin==3;
    c=0.302-0.577i; % 设置参数 c 的默认值
end
Nmax=100; % 设置最大迭代次数
M=100;    % 设置一个大数，表示无穷大
[x1,y1]=meshgrid(linspace(x-L,x+L,512),linspace(y-L,y+L,512)); % 计算采样点坐标值
N=ones(size(x1))*Nmax; % 设置 N 的初值
z=x1+y1*i;  % 生成复数坐标点
for k=1:Nmax;
    z=z.^2+c;       % 迭代计算新的 z 值
    N(abs(z)>M)=k; % 找出模值超过 M 的点
end
I1=image(linspace(x-L,x+L,512),linspace(y-L,y+L,512),N); % 根据 N 的数值绘图
title(['{\itc}=',num2str(c)]);
set(I1,'ButtonDownFcn',['p=get(gca,''currentpoint''),Julia(p(1),p(3),L,c);']);
%设置鼠标按下的相应内容
```



这里增加了函数 image 绘制图形时鼠标按下的相应内容。当按下鼠标左键或者右键时，属性 ButtonDownFcn 里面的语句将被执行。

制作用户界面窗口的程序如下：

```
c=0.302-0.577i;L=0.5;x=0;y=0; % 对参数赋默认值
figure('MenuBar','Figure','Position',[76 133 541 417]);     % 设置图形窗口的位置和大
小
set(gcf,'Name','Generation of Julia set','NumberTitle','off'); %设置图形窗口名称
A1=axes('Position',[0.1,0.1,0.6,0.7]);box on;      % 生成坐标轴
T1=uicontrol(gcf,'style','text','unit','normalized','position',...
   [0.24,0.9,0.04,0.05],'BackgroundColor',[0.9 0.9 0.9],...
   'ForegroundColor','b','fontsize',10,'string','c = '); % 标注文字
E2=uicontrol(gcf,'style','edit','unit','normalized','position',...

[0.29,0.9,0.18,0.05],'BackgroundColor','w','ForegroundColor','b','fontsize',10
,...
   'Callback','c=get(E2,''String'');c=str2num(c);Julia(p(1),p(3),L,c);'); % 生成
文本输入框
set(A1,'ButtonDownFcn',['p=get(A1,''currentpoint''),',...
      'Julia(p(1),p(3),L,c);axes(A1);']); % 设置坐标轴属性
S1=uicontrol(gcf,'style','slider','unit','normalized','position',...

[0.8,0.1,0.03,0.8],'BackgroundColor','w','ForegroundColor','b','fontsize',10,.
..
   'SliderStep',[0.01,0.01],'TooltipString','L=0','Value',0.5,'Callback',...

['L=get(S1,''Value'');set(S1,''TooltipString'',[''L='',num2str(L)]);Julia(p(1)
,p(3),L,c);']); % 生成滑动条用于控制 L 的数值
```

在如图 21.7 所示的坐标轴内单击一点就能以该点的横纵坐标为中心绘制 Julia 集（如图 21.8 所示）。在坐标轴上面的编辑框中可以输入参数 c 的取值。通过滑动条可以改变参数 L 的数值。

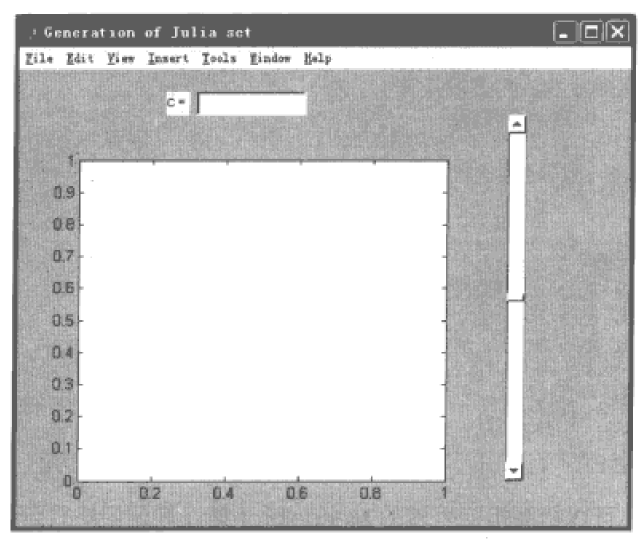


图 21.7 制作 Julia 集的图形界面窗口

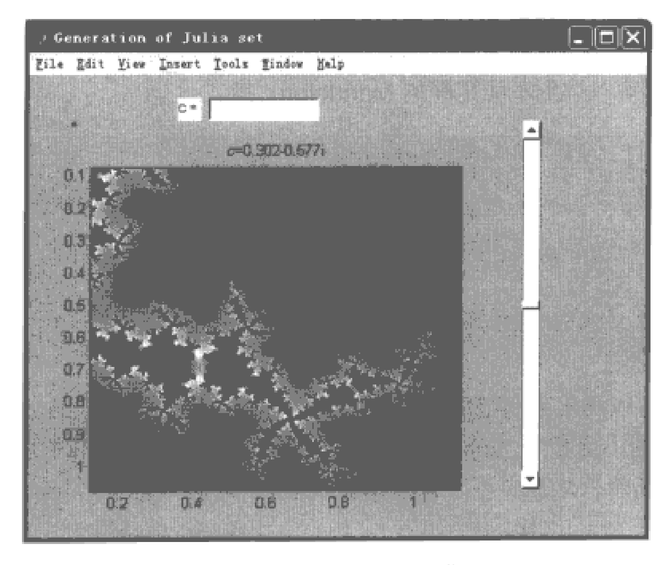


图 21.8 生成的 Julia 集

与 Julia 集相似的是 Mandelbrot 集，Mandelbrot 集的生成过程也是通过迭代关系 $z = z^2 + c$，不同的是 z 和 c 都是变化的，它们在相同的范围内取值。当某次（比如 k）迭代时，某点 z 的模值大于某个数 M，即把这个位置的函数值用迭代次数 k 代替，进行有限次迭代，可以得到一个关于迭代次数的函数，这个函数对应的图形就是 Mandelbrot 集。这里取参数 M 等于 2，z 的迭代初值统一取值为 1。当 z 的模值大于 M 时把对应位置的 c 和 z 值设置为 0，这样在迭代中这些已经达到 M 的点数值将始终等于 0（不可能再次达到 M）。下面给出 Mandelbrot 集的绘制程序。

```
function M_set(x,y,L);
%生成 mandelbrot 分形图
if nargin==3;
   c=0.302-0.577i; % 设置参数 c 的默认值
end
```

```
[x1,y1]=meshgrid(linspace(x-L,x+L,512),linspace(y-L,y+L,512)); % 在指定范围内离散采样
c=x1+i*y1;        % 生成复数 c 对应的矩阵
z=zeros(size(c));  % 设置 z 的初值
N=100; % 最大迭代次数
S=ones(size(c))*N; % 生成记录迭代次数 k 的矩阵，其初值设置为最大迭代次数
M=2; % 对参数 M 赋值
for k=1:N;
    z=z.^2+c; % 进行迭代
    S(abs(z)>2)=k; % 找出 z 的模值大于 M 的位置并赋值为 k
    c(abs(z)>2)=0; % 把 z 的模值大于 M 的位置相应的 c 值设置为 0
    z(abs(z)>2)=0; % 把 z 的模值大于 M 的位置相应的 c 值设置为 0
end
II=image(linspace(x-L,x+L,512),linspace(y-L,y+L,512),S); % 绘图
set(II,'ButtonDownFcn',['p=get(gca,''currentpoint''),M_set(x,y,L);']); % 设置鼠标按下的相应内容
```

上面的函数 M_set 可以绘制指定区域的 Mandelbrot 集，其调用格式为：

```
M_set(x,y,L);
```

参数说明： x 和 y 表示考虑范围的中心点横纵坐标。L 是方形区域的半边长。

利用下面一段程序可以得到不同范围的 Mandelbrot 集。

```
subplot(221);M_set(0,0,2);axis square;xlabel('(a)');          % 绘制 mandelbrot 集的图案
subplot(222);M_set(0.1,-0.8,0.5);axis square;xlabel('(b)');     % 绘制 mandelbrot 集的图案
subplot(223);M_set(0.15,-0.6,0.125);axis square;xlabel('(c)');  % 绘制 mandelbrot 集的图案
subplot(224);M_set(0.145,0.65,0.03);axis square;xlabel('(d)');  % 绘制 mandelbrot 集的图案
```

前面定义 M_set 可以画出不同范围内的 Mandelbrot 集，这里即调用函数 M_set 绘制不同范围内的 Mandelbrot 集。缩小函数 M_set 的输入参数 L，可以得到局部区域的放大效果。执行上述程序输出图形如图 21.9 所示，从图中可以看出 Mandelbrot 集可以得到非常丰富的图案。

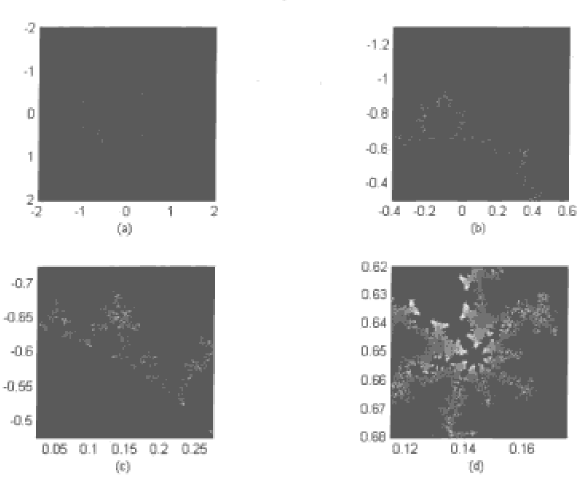


图 21.9　不同范围内的 Mandelbrot 集图形

类似于前面介绍的 Julia 集，可以制作一个用户界面来生成不同范围内的 Mandelbrot 集。通过界面窗口显示其中 Mandelbrot 集的细节部分，可以得到不同样式的美丽图案。

21.1.3 Koch 曲线

美国生物学家 Aristid Lindenmayer 在研究植物形态的进化与构造时，于 1968 年提出了一种描述方法，现在把这种方法称为 L 系统。1984 年 Smith 首次将 L 系统引入到计算机图形学领域。

L 系统是使用几个特殊符号描述分形图形的描画过程，然后把这些字符转化为分形图形。其建立字符串的过程是，设定替代字符串的单元及替换规则，然后对其中的某些字符串按照给定的规则进行替换。下面介绍一些特殊字符对应的意义。

F：以当前方向前进一步，并画线。

f：以当前方向前进一步，不画线。

+：逆时针旋转 δ 角度。

-：顺时针旋转 δ 角度。

[：将当前信息压栈。

]：把状态信息以“剪切”的方式从栈中取出。

ω：初始规则。

R：替换规则。

上面介绍的是一般的 L 系统的定义。这里考虑 Koch 曲线对应的 L 系统，具体可以写为：

ω: F

$\delta = 60^\circ$

R: F→F+F--F+F

首先把初始规则 ω 中的 F 按照替换规则 R 换为 F+F--F+F，得到一个新的字符串 ω_1，然后按替换规则替换字符串 ω_1 中的 F，得到 ω_2，依此类推，计算到我们设定的步数。最后我们根据最终的字符串 ω_n，按照每个字符的意义把 ω_n 转化为分形图。

下面给出利用 L 系统的方法绘制 Koch 曲线的例子。

```
omega='F'; % 初值规则
R='F+F--F+F'; % 替换规则
delta=pi/3;   % 旋转角度
z=i;    % 初始位置
A=0;  % 初始时的角度
N=4;  % 替换规则的次数
for k=1:N;
   omega=strrep(omega,'F',R); % 用 R 对符号 F 进行替换
end
hold on;
for k=1:length(omega);
   if strcmp(omega(k),'F');
      plot([z,z+exp(i*A)],'k'); % 如果遇到字符串 F 则画一线段
      z=z+exp(i*A);
   elseif strcmp(omega(k),'+');
      A=A+delta; % 更换角度值
   elseif strcmp(omega(k),'-');
      A=A-delta; % 更换角度值
   end
```

```
end
axis equal; box on
```

首先根据初始规则和替换规则得到最终的字符串，然后按照最终字符串中每个字符的含义进行绘图。执行程序得到的图形如图 21.10 所示，Koch 曲线的自相似性很容易看出来。

为方便起见，把上面程序中根据字符串绘图部分的程序保存为 Ls_draw 文件（其保存在光盘的\Ch21 文件夹下），在得到规则的字符串后可以直接调用函数 Ls_draw 来画图，该函数调用格式为：

```
Ls_draw(z,A, omega, delta);
```

参数说明：z 是初始时绘图起点的坐标，其为一个复数值。A 是初始时的角度。omega 是规则对应的字符串。delta 是旋转角度。

如果把初始时的规则写为“F--F--F--”，则可以得到 Koch 雪花，相应程序如下：

```
omega='F--F--F--'; % 初值规则
R='F+F--F+F'; % 替换规则
delta=pi/3;   % 旋转角度
z=i;  % 初始位置
A=0; % 初始时的角度
N=4; % 替换规则的次数
for k=1:N;
    omega=strrep(omega,'F',R); % 用 R 对符号 F 进行替换
end
Ls_draw(z,A,omega,delta); % 根据规则绘图
```

这里以一个封闭的三角形（即规则'F--F--F--'）作为 Koch 雪花的初始输入规则，然后多次利用替换规则得到一个字符串，通过调用函数 Ls_draw 对所得字符串绘图而得到 Koch 雪花的图案。输出图形如图 21.11 所示，可以看出 Koch 曲线就是由 3 个 Koch 曲线按一定的角度组合而成的。下面给出 L 系统所使用的不同规则，如表 21.1 所示。

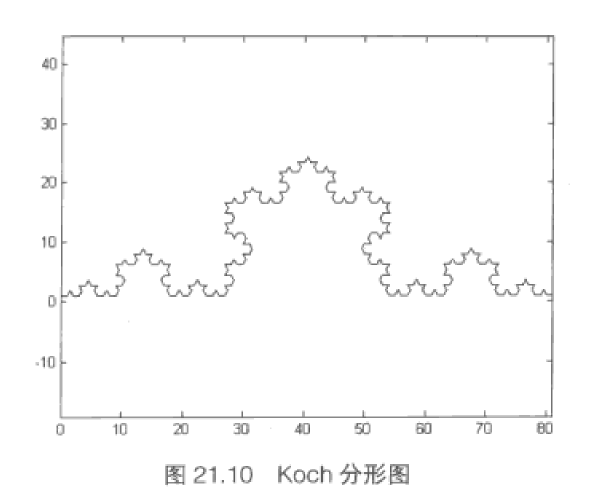


图 21.10　Koch 分形图

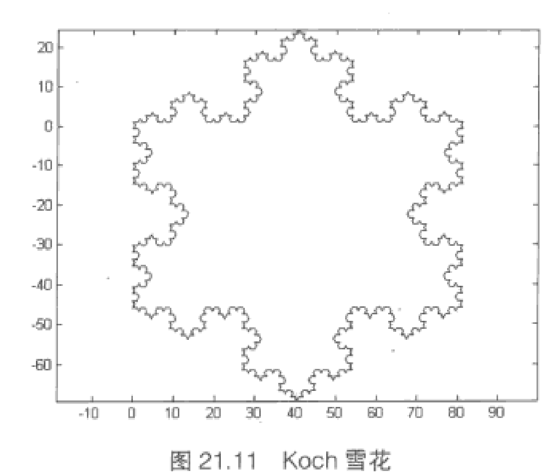


图 21.11　Koch 雪花

表 21.1　L 系统的规则

编号	旋转角度 δ	初始字符串 ω	替换规则 R
A	60°	F++F++F	F+F--F+F
B	90°	F-F-F-F	F-F+F+FFF-F-F+F

（续表）

编号	旋转角度δ	初始字符串ω	替换规则R
C	90°	F+F+F+F+	FF+F+F+F+F+F-F
D	90°	F-F-F-F	FF-F-F-F-FF

相应的绘图程序在光盘\Ch21 文件夹中的 Ls_draw_test.m 文件中，生成的图形如图 21.12 所示。读者可以设计其他的规则，利用 L 系统绘制相应的分形图形。

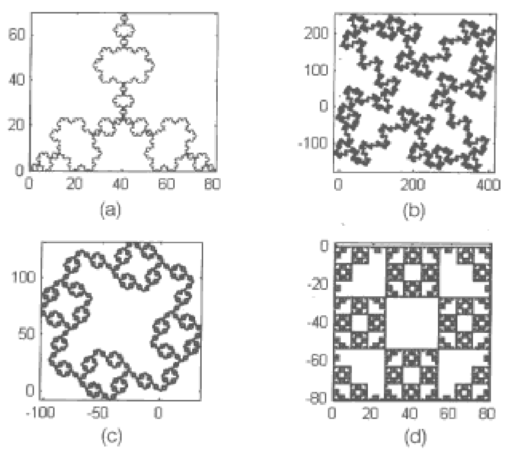


图 21.12　L 系统绘制的分形图形

21.2　迭代函数系统

本章前面介绍的分形图形都是使用线段作为基本单元绘制而成的，本节的方法主要是用“点”把分形图“点”出来，所用的方法叫做迭代函数系统。

21.2.1　基本定义

迭代函数系统（Iterated Function System，缩写为 IFS）是根据仿射变换而来的，其中仿射变换的数学表达式为：

$$\omega:\begin{cases}x'=ax+cy+e\\y'=bx+dy+f\end{cases}\quad 或\,\omega:\begin{bmatrix}x'\\y'\end{bmatrix}=\begin{bmatrix}a & c\\b & d\end{bmatrix}\begin{bmatrix}x\\y\end{bmatrix}+\begin{bmatrix}e\\f\end{bmatrix}=T\begin{bmatrix}x\\y\end{bmatrix}+\begin{bmatrix}e\\f\end{bmatrix}\tag{21-3}$$

其中x和y是变换前的坐标，x'和y'是变换后的坐标，a，b，c，d，e和f是变换的系数。这里用矩阵的形式便于计算。在变换表达式里面体现了缩放、旋转和平移，如果想绘制某个特殊的分形，读者应该找出分形的自相似特点，通过相似性确定仿射变换的系数。

为了绘制复杂的图形，需要多个仿射变换，即仿射变换集$\{\omega_n\}$。对应于每个仿射变换在绘图中都有一个固定的被选中几率，被选中则被调用。

以 Sierpinski 垫片为例，确定它的仿射变换的系数。在 Sierpinski 垫片中主要由 3 个部分作递归而得到，所以需要建立 3 个仿射变换$\{\omega_1,\omega_2,\omega_3\}$。同时 3 个部分的形状是完全一样的，只是当前层的边长等于前一层边长的一半，所以T为：

$$T=\begin{bmatrix}0.5 & 0\\0 & 0.5\end{bmatrix}$$

而 3 个部分的中心位置可以这样表示：$(0,0)$，$(0,1)$ 和 $\left(0.5,\sqrt{3}/2\right)$，这 3 点表示平移量。所以对应的 3 个仿射变换的系数和几率的数值如表 21.2 所示，相应的 MATLAB 程序如下。

```
N=20000; % 迭代次数
p=1/3;   % 设置几率值
q=2/3;   % 设置几率值
e=[1,0];  % 常数项
f=[0.5,sqrt(3)/2]; % 常数项
pxy=zeros(N,2); % 初始化坐标数据
r=rand(1,N-1);  % 生成随机数
for k=1:N-1;
    if r(k)<p
        pxy(k+1,:)=pxy(k,:)/2; % 按迭代式生成
    elseif r(k)<q;
        pxy(k+1,:)=pxy(k,:)/2+e; % 按迭代式生成
    else
        pxy(k+1,:)=pxy(k,:)/2+f; % 按迭代式生成
    end
end
P=complex(pxy(:,1),pxy(:,2)); % 把数据转化为复数
plot(P,'k.','markersize',2); % 绘图
```

根据表 21.2 进行多次循环计算，在每一步循环中根据随机数取值的大小选择计算式计算影射点。通过多次循环得到一组数据，然后把数据以点的方式绘制出来就可以得到 Sierpinski 垫片。

表 21.2　Sierpinski 垫片的 IFS 参数

i	a(i)	b(i)	c(i)	d(i)	e(i)	f(i)	p(i)
1	0.5	0	0	0.5	0	0	1/3
2	0.5	0	0	0.5	1	0	1/3
3	0.5	0	0	0.5	0.5	sin(pi/3)	1/3

执行上述程序输出图形如图 21.13 所示，尽管是一个随机过程，但最后还是得到了规则的图形，其中的规律可以认为是：初始一个涂黑的正三角形，每次切去正三角形中间的一个小正三角形，多次下去而得到了 Sierpinski 垫片。

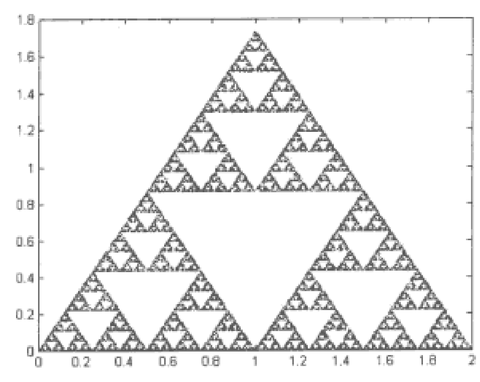


图 21.13　Sierpinski 垫片图案



在迭代函数系统中的迭代次数N，读者还可以改变它的数值，当N取值较大时数据点的数目多，图形会变得清晰。

下面给出利用迭代函数系统绘制Koch曲线的方法，相关参数如表21.3所示。

表21.3 Koch曲线的迭代函数系统参数

i	a	b	c	d	e	f	p
1	1/3	0	0	1/3	0	0	1/4
2	1/6	1/sqrt(12)	-1/sqrt(12)	1/6	1/3	0	1/4
3	1/6	-1/sqrt(12)	1/sqrt(12)	1/6	1/2	1/sqrt(12)	1/4
4	1/3	0	0	1/3	2/3	0	1/4

下面给出利用迭代函数系统绘制分形图形的一个函数，其内容如下：

```
function IFS_draw(M,p);
N=30000;  % 迭代次数
for k=1:length(p); % 生成映射矩阵a1,a2,...
   eval(['a',num2str(k),'=reshape(M(',num2str(k),',:),2,3);']);
end
xy=zeros(2,N);    % 设置迭代矩阵初值
pp=meshgrid(p); % 生成网格矩阵
pp=tril(pp);      % 取上三角矩阵
pp=sum(pp,2); % 对行向量求和，从而得到几率的累加结果
for k=1:N-1;
   a=rand-pp;    % 计算随机数和几率向量的差值
   d=find(a<=0);% 找出满足几率条件的位置

xy(:,k+1)=eval(['a',num2str(d(1)),'(:,1:2)'])*xy(:,k)+eval(['a',num2str(d(1)),
'(:,3)']); % 计算迭代点
end
P=complex(xy(1,:),xy(2,:)); % 生成复数点
plot(P,'k.','markersize',2);  % 画图
axis equal; % 设置坐标轴属性
```

函数IFS_draw的调用格式为：

```
IFS_draw(M,p);
```

参数说明： M为映射矩阵的系数以及常数项部分组成的矩阵，如表21.3中数值阵列对应的矩阵。p是一个几率向量，其顺序应该和矩阵M中映射输入顺序一致。

下面利用函数IFS_draw来绘制Koch曲线，相关的语句为：

```
M=[1/3,0,0,1/3,0,0;...
     1/6,1/sqrt(12),-1/sqrt(12),1/6,1/3,0;...
     1/6,-1/sqrt(12),1/sqrt(12),1/6,1/2,1/sqrt(12);...
     1/3,0,0,1/3,2/3,0]; % 迭代矩阵M
p=ones(1,4)/4;  % 几率向量
IFS_draw(M,p); % 调用函数绘图
```

使用迭代函数系统绘制Koch曲线时，先定义映射矩阵系数矩阵M，同时定义对应的几率向量p，然后调用函数IFS_darw绘制Koch曲线。执行上述程序输出图形如图21.14所示，其图案与前

面利用 L 系统绘制的 Koch 曲线很相似。

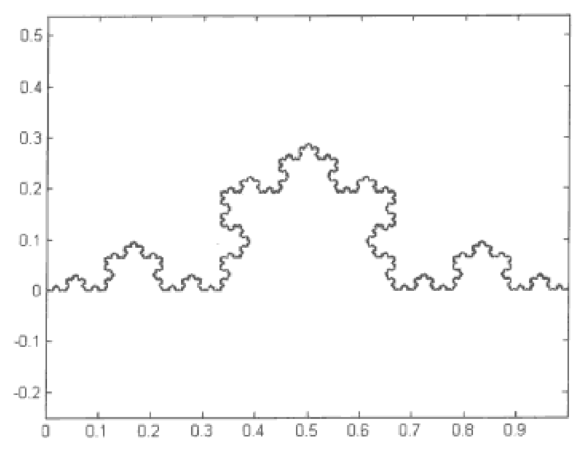


图 21.14　利用迭代函数系统绘制的 Koch 曲线

下面给出一个六角雪花的迭代函数参数，如表 21.4 所示。

表 21.4　六角雪花的迭代函数系统参数

i	a	b	c	d	e	f	p
1	1/3	0	0	1/3	cos(pi/3)	sin(pi/3)	1/6
2	1/3	0	0	1/3	cos(2*pi/3)	sin(2*pi/3)	1/6
3	1/3	0	0	1/3	cos(pi)	sin(pi)	1/6
4	1/3	0	0	1/3	cos(4*pi/3)	sin(4*pi/3)	1/6
5	1/3	0	0	1/3	cos(5*pi/3)	sin(5*pi/3)	1/6
6	1/3	0	0	1/3	1	0	1/6

```
a=1/3; % 比例系数
A=pi/3; % 角度间隔
M=[a,0,0,a,cos(A),sin(A);...
    a,0,0,a,cos(2*A),sin(2*A);...
    a,0,0,a,cos(3*A),sin(3*A);...
    a,0,0,a,cos(4*A),sin(4*A);...
    a,0,0,a,cos(5*A),sin(5*A);...
    a,0,0,a,1,0]; % 迭代矩阵
p=ones(1,6)/6;  % 几率向量
IFS_draw(M,p); % 调用函数绘图
axis image
set(gcf,'Color','w');
```

使用迭代函数系统绘制六角雪花时，首先定义映射矩阵系数（其在表 21.4 中给出）矩阵 M，同时定义对应的几率向量 p（其数据在表 21.4 最后一列），然后调用函数 IFS_darw 绘制六角雪花。执行程序输出图形如图 21.15 所示，从中可以看出分形图形的自相似性特点。

21.2.2　分形树叶

下面给出两个利用迭代函数系统绘制树叶的例子。第 1 片树叶迭代函数系统的参数如表 21.5 所示。

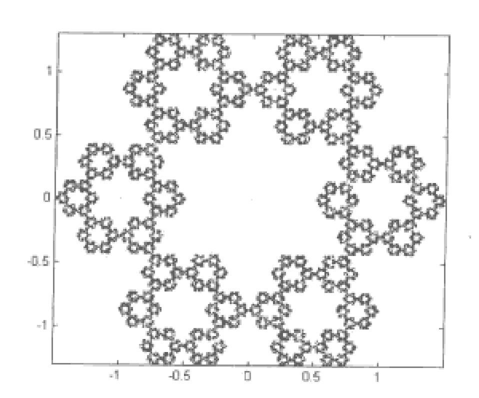


图 21.15 利用迭代函数系统生成的雪花图案

表 21.5 树叶 1 迭代函数系统的参数

编号	a	b	c	d	e	f
1	0.65	0	0	0.65	0.2	0.4
2	0.55	0	0	0.55	0.2	0.135
3	0.38	-0.28	0.28	0.38	0.3	0.4
4	0.38	0.28	-0.28	0.38	0.3	0.1

其中所有几率为 25。

为了得到更清晰的图案，在绘制分形树叶的时候把 IFS_draw.m 文件中的迭代次数 N 改为 300000，其他参数不变。相应的 MATLAB 程序如下：

```
M=[0.65,0,0,0.65,0.2,0.4;...
    0.55,0,0,0.55,0.2,0.135;...
    0.38,-0.28,0.28,0.38,0.3,0.4;...
    0.38,0.28,-0.28,0.38,0.3,0.1]; % 生成系数矩阵 M
p=ones(1,4)/4;    % 生成几率向量
IFS_draw(M,p);  % 调用函数绘图
axis image;      % 设置坐标轴属性
```

与前面绘制 Koch 曲线和六角雪花相似，首先定义映射矩阵系数（其在表 21.5 中给出）矩阵 M，同时定义对应的几率向量 p（其数据在表 21.5 最后一列），然后调用函数 IFS_darw 绘制分形树叶。执行上述程序所得的树叶图形如图 21.16 所示，该图案整体看上去很像树叶。

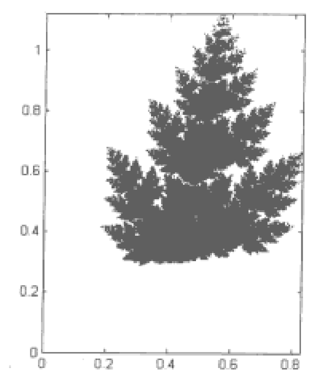


图 21.16 利用迭代函数系统生成的树叶

下面给出另一种树叶的迭代函数系统的参数，如表 21.6 所示。

表 21.6　树叶 2 迭代函数系统的参数

编号	a	b	c	d	e	f	p
1	0	0	0	0.15	0	0	0.08
2	0.83	-0.02	0.04	0.83	0	2	0.8
3	0.2	0.22	-0.3	0.3	0	2	0.096
4	-0.14	0.24	0.31	0.27	0	0.5	0.096

具体实现程序如下：

```
M=[0,0,0,0.15,0,0.2;...
   0.83,-0.02,0.04,0.83,0,2;...
   0.2,0.22,-0.3,0.3,0,2;...
   -0.14,0.24,0.31,0.27,0,0.5]; % 生成系数矩阵 M
p=[0.08,0.8,0.096,0.096];      % 生成几率向量
IFS_draw(M,p); % 调用函数绘图
axis image;    % 设置坐标轴属性
```

执行上述程序所得的图形如图 21.17 所示。与图 21.16 不同的是，在图 21.17 中含有叶柄。

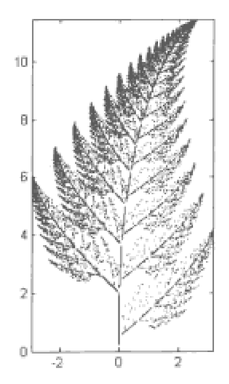


图 21.17　迭代函数系统生成的树叶

21.2.3　分形树

本小节介绍利用迭代函数方法绘制树木图案的实例。树木 1 的迭代函数系统参数如表 21.7 所示。

表 21.7　树木 1 的迭代函数系统参数

编号	a	b	c	d	e	f	p
1	-0.64	0	0	0.5	0.86	0.25	0.06
2	-0.04	-0.47	0.07	-0.02	0.49	0.51	0.22
3	0.2	0.33	-0.49	0.43	0.44	0.25	0.23
4	0.46	-0.25	0.41	0.36	0.25	0.57	0.24
5	-0.06	0.45	-0.07	-0.11	0.59	0.1	0.25

相应的 MATLAB 程序如下：

```
M=[-0.64,0,0,0.5,0.86,0.25;...
    -0.04,-0.47,0.07,-0.02,0.49,0.51;...
    0.2,0.33,-0.49,0.43,0.44,0.25;...
    0.46,-0.25,0.41,0.36,0.25,0.57;...
    -0.06,0.45,-0.07,-0.11,0.59,0.1]; % 生成系数矩阵 M
p=[0.06,0.22,0.23,0.24,0.25]; % 生成几率向量
IFS_draw(M,p); % 调用函数绘图
axis image; % 设置坐标轴属性
```

与前面绘制 Koch 曲线和六角雪花相似，首先定义映射矩阵系数（其在表 21.7 中给出）矩阵 M 和几率向量 p（其数据在表 21.7 最后一列），然后调用函数 IFS_darw 绘制树木图案。执行上述程序输出图形如图 21.18 所示，该图很像春天刚刚发芽的树木。

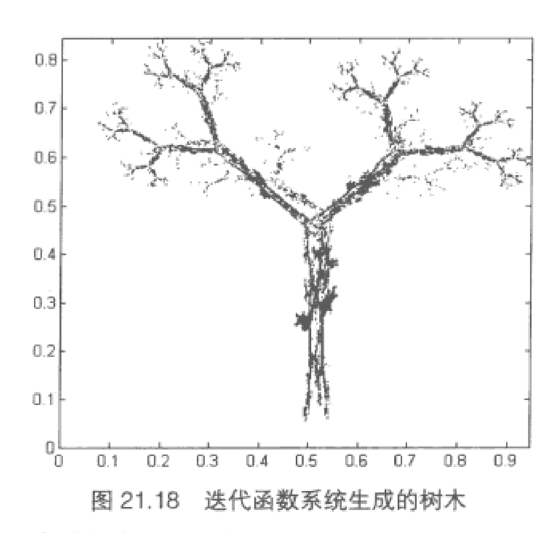


图 21.18 迭代函数系统生成的树木

树木 2 的迭代函数系统参数如表 21.8 所示。

表 21.8 树木 2 的迭代函数系统参数

编号	a	b	c	d	e	f	p
1	0.06	0	0	0.6	0	0	0.1
2	0.04	0	0	-0.5	0	1	0.1
3	0.46	-0.34	0.32	0.38	0	0.6	0.1
4	0.48	0.17	-0.15	0.42	0	1	0.23
5	0.43	-0.26	0.27	0.48	0	1	0.23
6	0.42	0.35	-0.36	0.31	0	0.8	0.24

相应 MATLAB 程序如下：

```
M=[0.06,0,0,0.6,0,0;...
       0.04,0,0,-0.5,0,1;...
       0.46,-0.34,0.32,0.38,0,0.6;...
       0.48,0.17,-0.15,0.42,0,1;...
       0.43,-0.26,0.27,0.48,0,1;...
       0.42,0.35,-0.36,0.31,0,0.8]; % 生成系数矩阵 M
p=[0.1,0.1,0.1,0.23,0.23,0.24];       % 生成几率向量
IFS_draw(M,p); % 调用函数绘图
axis image;    % 设置坐标轴属性
```

与前面绘制 Koch 曲线和六角雪花相似，首先定义映射矩阵系数（其在表 21.8 中给出）矩阵 M 和几率向量 p（其数据在表 21.8 最后一列），然后调用函数 IFS_darw 绘制树木图案。执行上述程序输出如图 21.19 所示的图形，该图案很想一株夏天的树木，长有茂密的树叶。

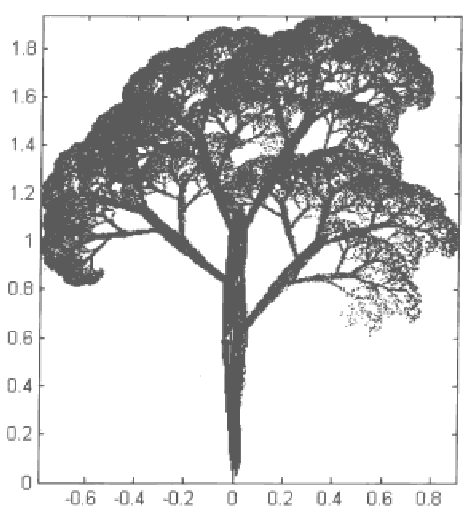


图 21.19　利用迭代函数系统绘制的树木

21.2.4　龙曲线

本小节来讨论外形像“龙”的分形图案，它们可以利用 IFS 绘制。下面给出龙曲线的迭代函数系统的参数，如表 21.9 所示。

表 21.9　龙曲线的迭代函数系统参数

编号	a	b	c	d	e	f
1	0.5	0.5	-0.5	0.5	1	0
2	0.5	0.5	-0.5	0.5	-1	0

其中编号 1 和编号 2 的几率都是 50%。相应的 MATLAB 程序如下：

```
M=[0.5,0.5,-0.5,0.5,1,0;...
      0.5,0.5,-0.5,0.5,-1,0]; % 生成系数矩阵 M
p=[0.5,0.5];      % 生成几率向量
IFS_draw(M,p); % 调用函数绘图
axis image;      % 设置坐标轴属性
```

与前面绘制 Koch 曲线和六角雪花相似，首先定义映射矩阵系数（其在表 21.9 中给出）矩阵 M 和几率向量 p（其数据在表 21.9 最后一列），然后调用函数 IFS_darw 绘制龙曲线图案。输出的图形如图 21.20 所示，图案看起来像长有多只头的龙。

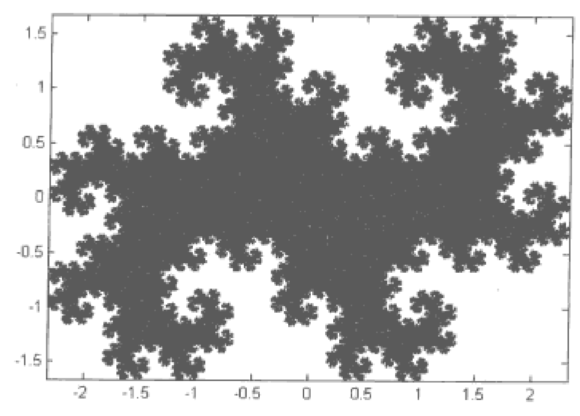


图 21.20 利用迭代函数系统绘制的龙曲线图

下面再给出一个龙曲线的例子，相关的参数如表 21.10 所示。

表 21.10 龙曲线的迭代函数系统参数

编号	a	b	c	d	e	f	p
1	0.8	-0.2	0.3	0.9	-1.9	-0.1	0.7
2	0.1	-0.5	0.5	-0.4	0.8	7	0.3

相应的 MATLAB 程序如下：

```
M=[0.8,-0.2,0.3,0.9,-1.9,-0.1;...
      0.1,-0.5,0.5,-0.4,0.8,7]; % 生成系数矩阵 M
p=[0.7,0.3];    % 生成几率向量
IFS_draw(M,p); % 调用函数绘图
axis image;    % 设置坐标轴属性
```

该迭代函数系统只有两个映射关系条件。与前面绘制 Koch 曲线和六角雪花相似，首先定义映射矩阵系数（其在表 21.10 中给出）矩阵 M 和几率向量 p（其数据在表 21.10 最后一列），然后调用函数 IFS_darw 绘制龙曲线图案。执行上述程序输出图形如图 21.21 所示，该图案很像一条扭曲的龙。

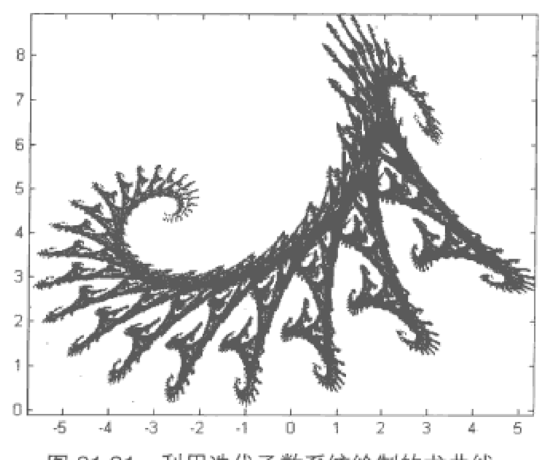


图 21.21 利用迭代函数系统绘制的龙曲线

通过前面的介绍，读者可以发现利用迭代函数系统可以绘制不同类型的分形图案。迭代函数系统中的映射关系就如同画家脑海中的思绪，利用这些映射随机迭代就可以用 MATLAB 程序绘制出美妙的图案。读者可以自己设计一些映射关系来绘制丰富的图案。在使用迭代函数系统时，虽然迭代函数系统的计算量比前面提到的递归算法和 L 系统大，但是它的时间主要是花在计算数据上，相对绘图的时间较短，计算后使用 plot 函数一次就可以了。在迭代过程中，牺牲了计算量，但换取了对分形图细节的描绘，使得分形图更加逼真。

21.3 递归算法

前面介绍了迭代算法绘制分形图的方法，本节介绍利用递归算法绘制分形图。分形图形的主要特点是自相似性，而递归算法就是按某一规则进行嵌套来绘制图形。可以使用不同的基元图形来递归。递归的一般结构如下：

```
function recursion(n);
……
recursion(m);
……
```

实质上，递归算法是根据计算机中“压栈”和“出栈”的概念，重复地使用某规则来绘制图形。“压栈”是停止目前的操作，记录当前信息，再去完成更低一层的操作；“出栈”是结束在该层的操作回到更高的层，开始因为“压栈”而中断的操作。这样的程序结构是一种特殊的结构，可以大大缩短程序的长度。对于初次接触这个算法的人来说是比较难理解的。分形的自我相似、复制和嵌套，很容易和递归算法结合起来，很多的经典分形图形都可以使用递归算法计算。

21.3.1 分形树木

前面一节介绍了树木的迭代函数系统的画法，这里给出利用递归方法绘制树木的例子。通过几条线段的递归就可以得到一些图案酷似树或者树叶的图形，下面介绍一些画法。

用递归算法对线段 CD，CB，CE，BF 和 BG 用图 21.22 中的左图进行递归就可以得到右侧的图形，程序参见光盘中\Ch21 文件夹下的 fractals_tree_2.m 文件，在命令窗输入以下的代码，就可以得到图 21.22 右图所示的图形。

```
subplot(121)
[z1,z2]=fractals_tree_2(2+i,2+4i,1);
subplot(122)
[z1,z2]=fractals_tree_2(2+i,2+4i,5);
```

另外使用图 21.23 左侧的生成元进行递归就可以得到右侧图形，具体程序可以参阅光盘中\Ch21 文件夹下的 fractals_tree3.m 和 fractals_tree_3.m 程序文件。读者可以改变其中的角度，再观察结果的变化。

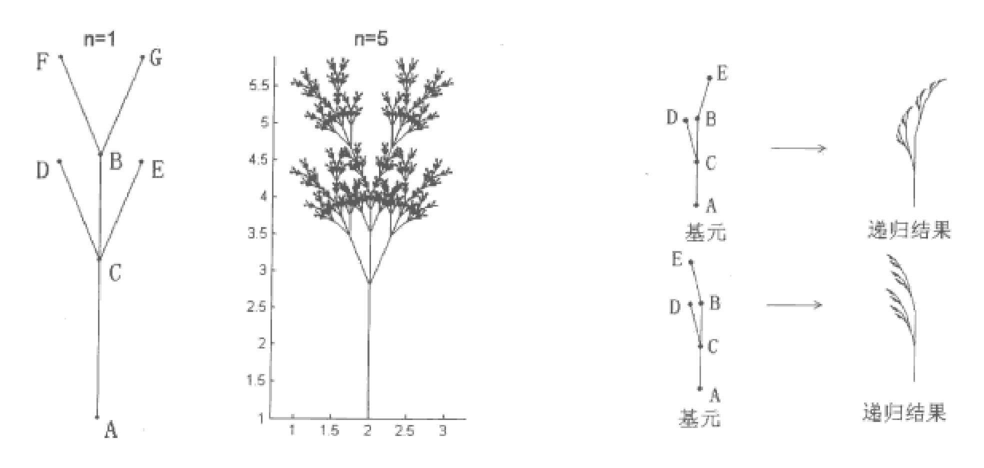


图 21.22　递归算法绘制的分形树　　　　图 21.23　分形树的生成元及其递归结果

在图 21.24 中，使用两个递归方式与图 21.22 不同就会得到完全不同的结果，此时的结果很像树叶。在图 21.24 中只是进行了 5 层递归，次数多一些能够更逼近树叶。读者可以参见光盘中\Ch21 文件夹下的 fractals_leaf.m 文件，对比和图 21.22 的不同之处。

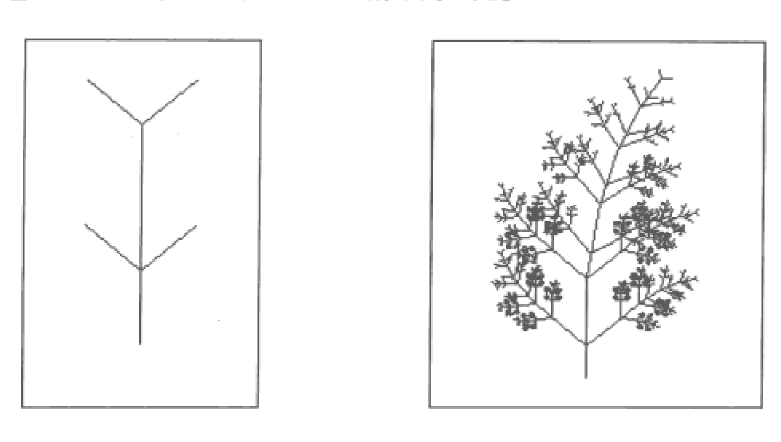


图 21.24　分形树叶

图 21.25 利用 3 个分叉进行递归，得到一棵伞状的树木，请参见光盘中\Ch21 文件夹下的 fractals_tree4.m 文件。

分形树和分形树叶的画法有很多，可以使用不同的生成元对其中部分或者全部线段进行递归，而且可以改变其中的角度和程度参数。当然也可以控制长度和角度数值为某一个范围内的随机数，这样产生的图形失去规则性，看上去就更接近于自然图形了。实际计算的时候递归的深度要合适，能够得到实际图形就行了，不要一味地增加递归深度，这样会耗费很多时间。当然选择合适的递归更好，使计算量尽可能少。这些都是技巧，可以在实际编程时慢慢地摸索。



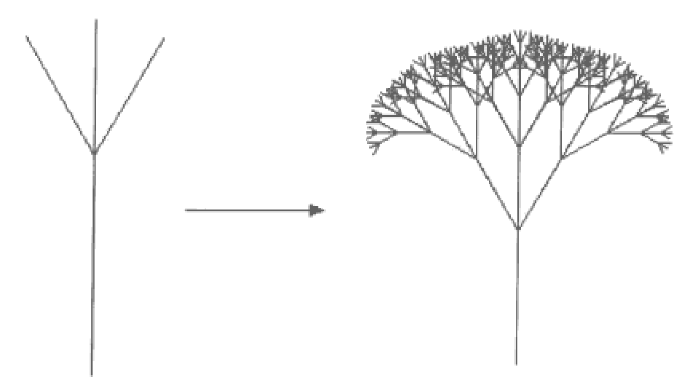


图 21.25　分形树

21.3.2 Arboresent 肺

Arboresent 肺也可看成是一种 Koch 曲线。初始是两个顶角为钝角的等腰三角形，它们有一个公共的锐角顶点（如图 21.26 中 n=0 的情况）。从这个顶点出发，两个等腰三角形的长边组成有一个夹角的两个等腰三角形，这两个等腰三角形与原来的三角形具有相似关系，相似比为三角形的腰的长度和底边长度的比值。实际上，上面的过程可以在 n=0 时，把两侧的三角形分别递归，这样有自相似关系，而 n=0 时，整体处理还是不具有递归关系的。可以参阅光盘中\Ch21 文件夹下的 arboresent_lung.m 文件。只要在命令窗输入下面的内容就可以得到如图 21.26 所示图形。

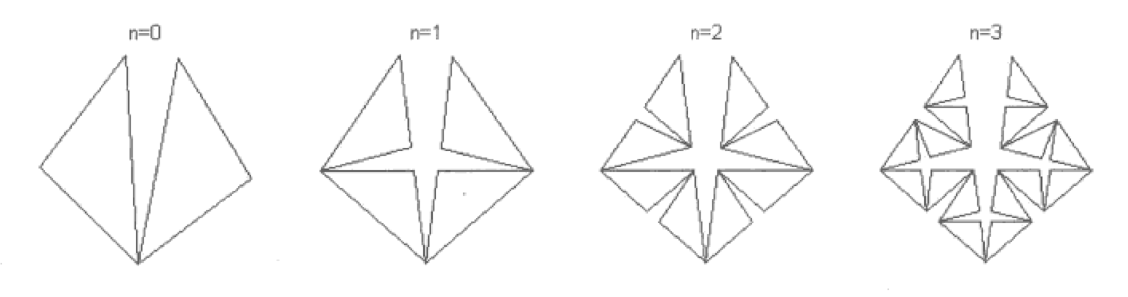


图 21.26　Arboresent 肺

```
for k=1:4
subplot(1,4,k);            % 生成子图
arboresent_lung(k-1);      % 调用函数 arboresent_lung 绘图
title(['n=',num2str(k-1)]);  % 添加图题
axis square image off;     % 设置坐标轴属性
end
```

可以增大 arboresent_lung 函数的输入参数 n，以得到更精细的结构。

21.3.3 Sierpinski 垫片

图 21.13 给出了等腰三角形的 Sierpinski 垫片，这里再给出正方形的 Sierpinski 垫片图案。Sierpinski 垫片也可以看成是 Koch 曲线的二维扩展。它的生成规则是：首先在平面上构造一个正方形，将正方形的每条边三等分（得到 9 个小正方形，如图 21.27 所示），把中间的正方形挖除，再把剩下的 8 个小正方形按照同样的方法进行分割与挖除。理论上此过程执行到无穷小的正方形

即得到 Sierpinski 垫片。实际上选择到最小正方形的边长是初始正方形的$1/3^4$，即 1/81 就可以了，再小下去视觉上就看不到了。递归算法步骤如下：

step 1 设定初始正方形左上角和右下角的坐标为(x_1, y_1)和(x_2, y_2)，则正方形的长宽分别为$L = x_2 - x_1$，$W = y_2 - y_1$（说明：考虑到计算方便才这样定义，此时W是负数），根据这两个点就可以绘制正方形了。

step 2 计算 9 个小正方形的左上角和右下角坐标，这里把结果以表格形式给出，如表 21.11 所示。

1	2	3
4	0	5
6	7	8

图 21.27　区域分割法

表 21.11　正方形的坐标值

正方形编号	左上角坐标	右下角坐标
0	$(x_1+L/3, y_1+W/3)$	$(x_2-L/3, y_2-W/3)$
1	(x_1, y_1)	$(x_1+L/3, y_1+W/3)$
2	$(x_1+L/3, y_1)$	$(x_2-L/3, y_1+W/3)$
3	$(x_2-L/3, y_1)$	$(x_2, y_1+W/3)$
4	$(x_1, y_1+W/3)$	$(x_1+L/3, y_2-W/3)$
5	$(x_2-L/3, y_1+W/3)$	$(x_2, y_2-W/3)$
6	$(x_1, y_2-W/3)$	$(x_1+L/3, y_2)$
7	$(x_1+L/3, y_2-W/3)$	$(x_2-L/3, y_2)$
8	$(x_2-L/3, y_2-W/3)$	(x_2, y_2)

step 3 设定递归深度，对编号为 1~8 的正方形作递归处理，就可以得到 Sierpinski 垫片的分形图形，如图 21.28 所示。为了书写简单，MATLAB 程序中使用W和L替换$W/3$和$L/3$。该算法对应的程序是光盘中\Ch21 文件夹下的 Sierpinski_carpet.m，其中W和L取相等数值是正方形，如果它们不相等就是长方形对应的 Sierpinski 垫片。

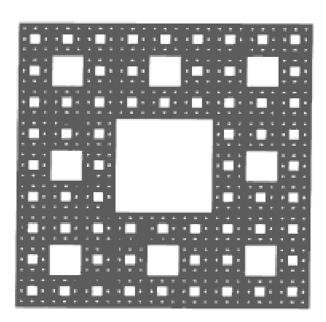


图 21.28　正方形的 Sierpinski 垫片

21.3.4 Peano 曲线

意大利的数学家 Peano 通过对一些古代装饰图案的研究，构造了一条奇怪的平面曲线。这条曲线蜿蜒曲折一气呵成，并能经过平面上某一正方形区域内“所有点”。当时引起数学界的广泛注意，不久研究者们便找到了具有这样性质的其他曲线，后来统称为 Peano 曲线。Peano 曲线的一个典型例子就是 Hilbert-Peano 曲线，如图 21.29 所示。

该曲线的生成元为三条正交的线段，图 21.30 给出了该曲线的递归过程示意图。具体递归步骤如下：

step 1 把正方形四等分，求出各小正方形的中心，将它们连接起来，可以看到存在两种连接方式，如图 21.31 所示。

step 2 将各个小的正方形再细分为 4 个相同的小正方形，并连接各正方形的中心，也将会发展成两种连接方式。

step 3 依此类推就可以得到相应的图形。对应程序可以参阅光盘中\Ch21 文件夹下的 Hilbert_Peano.m 文件。

图 21.29　Hilbert-Peano 曲线

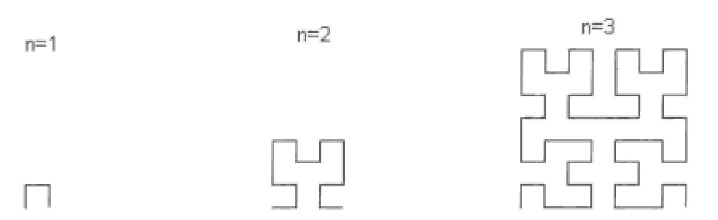


图 21.30　Hilbert-Peano 曲线的递归过程

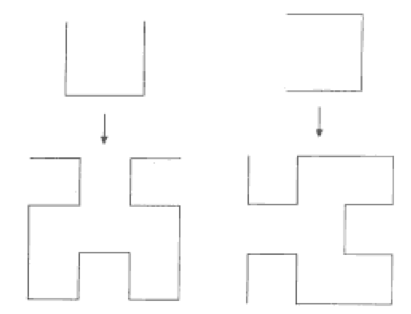


图 21.31 Hilbert-Peano 曲线生成示意图

此外，Hilbert-Peano 曲线还可以使用另外一种方法来计算。如图 21.32 所示，在 n=2 时的状态，如果把曲线分成 4 个部分，那么每一部分可以看成是由 n=1 时的曲线经过平移或者旋转操作得到的。比如说 n=2 时，A 区的曲线可以看成是 n=1 的曲线顺时针旋转 90° 而得到的；B 区曲线是 n=1 时曲线旋转 180° 再平移一段距离得到的；C 区的曲线和 D 区的曲线都是 n=1 时曲线平移操作得到的。可以验证这个规律对从 n=2 状态到 n=3 状态是否成立。根据这个规律我们可以很快写出程序。

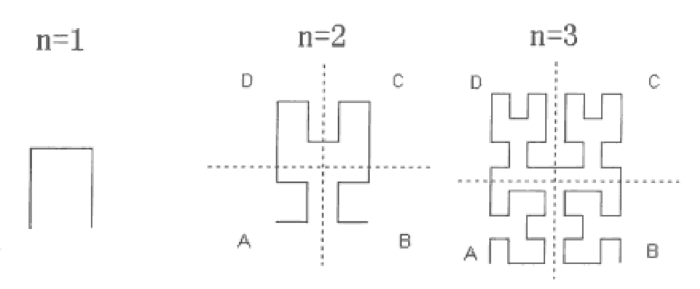


图 21.32 Hilbert-Peano 曲线生成过程示意图

为了计算方便，规定 n=1 时左下角处端点坐标在原点上，n=1 时每段线段的长度是 1。接下来按照上面说的规律进行递归处理就可以了。递归深度将决定坐标轴的范围，比如说，n=1 时 X 轴范围是[0,1]，n=3 时 X 轴的范围是[0,8]，长度就是 2 的 n 次幂。此算法对应的 MATLAB 程序是光盘中\Ch21 文件夹下的 Hilbert_Peano2.m 文件。

最后再介绍 Hilbert-Peano 曲线的一个生成程序，即 peano.m 文件（该文件保存于光盘中\Ch21 文件夹下），该文件的对应算法就不在这里描述了，感兴趣的读者可以自己研究一下。

21.3.5 C 曲线

如图 21.33 所示的图形非常像一个躺着的字母 C，所以称它为 C 曲线。其实它的原名叫 Levy 曲线，是法国数学家 Paul Levy 构造的。很明显，它具有很强的自相似性，也很漂亮，具有装饰性。它的生成规则非常简单，即从一条直线出发，然后向下支起一个 90° 的角，在角的两条边上再分别向外支起 90° 的角，依此类推，就可以得到 C 曲线。读者可以对照图 21.34 理解。

算法步骤如下：

step 1 先给定初始线段的端点，(x_1, y_1) 和 (x_2, y_2)，并连接它们。

step 2 计算向下支起的点的坐标：$x_3 = (x_1 + y_1 + x_2 - y_2)/2$，$y_3 = (x_1 - x_2 - y_2 - y_1)/2$。

从第三点分别和初始线段的端点连接。

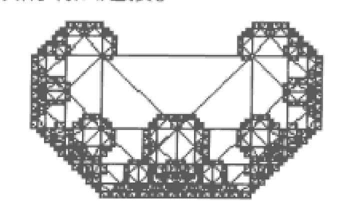


图 21.33　递归算法生成的 C 曲线

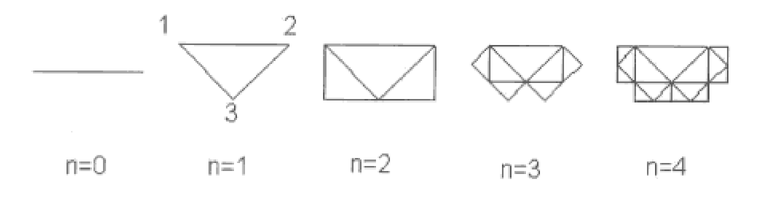


图 21.34　C 曲线的构造过程

step 3　以(x_1, y_1)和(x_3, y_3)为输入点，按步骤 1 和步骤 2 进行递归。

step 4　以(x_2, y_2)和(x_3, y_3)为输入点，按步骤 1 和步骤 2 进行递归。直至递归深度，完成制作的过程。

C 曲线实现程序是 drawC.m 文件（其保存于光盘的\Ch21 文件夹下），程序中为书写简单，使用复数表示点的坐标。本程序在递归深度取值较大（比如 n=10）时将要耗费很长的时间，读者可以使用较小的递归深度，比如 n=8，然后使用鼠标缩小 figure，就可以得到如图 21.33 所示的图案了。

21.3.6　多角星构成的分形图

如果以多角星作为单元进行递归计算可以得到不同的分形图案，下面给出以七角星为单元的递归图案的程序。

```
function [z,A]=recurs_star7(z,A,n);
hold on
nr=7;     % 旋转对称次数
N=4;     % 递归次数
sc=0.3;  % scale 缩小率
kr=4;     % 角度间隔数
ang=pi/nr/2; % 最小角度
rand('state',n+10) % 设置随机数状态
C=rand(1,3);      % 生成随机的颜色向量
if n<N;
   r=sc^n;  % 计算半径的长度
   [z,A,P]=star7(z,A,r);   % 调用自函数绘制七角星
   fill(real(P),imag(P),C); % 对七角星进行填充
   if abs(n-1)<1e-5;
      m=nr; % 确定角点的数目
      k=0;  % 计算角度初值指标 k
   else
      m=nr-1; % 确定角点的数目
      k=1;    % 计算角度初值指标 k
```

```
    end
    for p=1:m;
        A=A+pi; % 更新角度变量 A
        A=A+(p-1+k)*kr*ang;  % 计算角度
        z=z+r*(1+sc)*exp(i*A); % 获取下一位置的坐标，其为一复数
        [z,A]=recurs_star7(z,A,n+1); % 递归调用
        A=A+pi; % 变化角度值
        z=z+r*(1+sc)*exp(i*A); % 更新位置坐标值
        A=A-(p-1+k)*kr*ang;   % 计算角度值
    end
end
end
axis equal; % 设置坐标轴属性
function [z,A,P]=star7(z,A,r);
nr=7;       % 执行角点树目
ang=pi/nr/2; % 计算最小角度
ks=2*cos(pi/nr/2); % 计算缩小比例
A=A+pi;          % 更新角度值
z=z+r*exp(i*A);   % 计算顶点位置
A=A+pi-ang; % 计算初始时角度的数值
P=z;        % 初始化位置向量
for k=1:nr;
   plot([z,z+r*ks*exp(i*A)]); % 绘图
   z=z+r*ks*exp(i*A); % 计算下一位置的角度
   P=[P,z]; % 把 z 的数值添加到向量 P 中
   A=A+pi-2*ang;    % 更新角度值
end
A=A+ang;      % 变化 A 的数值
z=z+r*exp(i*A); % 改变 z 的数值
```

执行语句“[z,A]=recurs_star7(0,-pi/2,1);”后输出图形如图 21.35 所示。

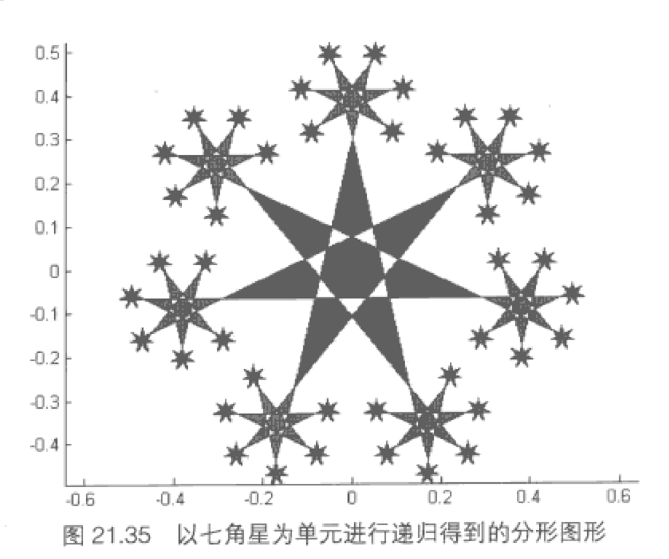


图 21.35　以七角星为单元进行递归得到的分形图形

如果 Koch 曲线的生成元是如图 21.36 所示的矩阵方框内的折线，对应的分形图可以参见图 21.36 下面的曲线。该图形的 MATLAB 程序可以参见光盘中\Ch21 文件夹下的 Koch_ Cantort.m 文件。

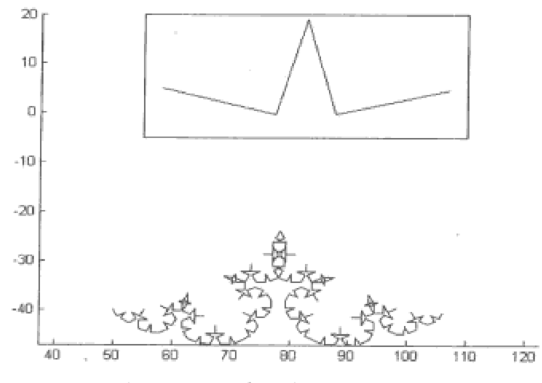


图 21.36　变形的 Koch 曲线

21.4　分维的计算

在几何学中，维数的概念非常重要，比如点、线、面、体的维数分别是 0，1，2，3。空间的维数常被理解为坐标数（或者是描述对象的最少独立变量数目），比如线性代数中的 n 维空间。对于分形图形来说，利用最少描述对象的独立变量数目就不合适了。在维数研究中，Hausdorff 引入的维数的概念是比较重要的进展。本节来介绍维数的定义与计算。

对于经典的分形，如 Cantor 集和 Koch 曲线，它们的局部形状和整体之间满足相似关系，即局部图形放大一定比例后与整体相同，这类图形被称为自相似集。对于这类图形，维数的定义是：

$$D_s = \frac{\ln m}{\ln(1/c)} \tag{21-4}$$

其中 D_s 是自相似集图形的维数，被称为相似维数。m 是当前尺度下的一个单元包含下一尺度下的子单元数目。c 是相似比。对于 Cantor 集，$m=2$，$c=1/3$，$D_s=\ln 2/\ln 3=0.6309$；而在 Koch 曲线中，$m=4$，$c=1/3$，$Ds=\ln 4/\ln 3=1.2619$。

对于复杂的分形图形，无法确定参数 m 和 c，从而不能计算出对应的相似维数 D_s。此时需要考虑其他方法计算图形的维数。可以设想使用一些大小相同的方形框或者方块（其边长为 r）来覆盖被测量的图形对象。通过统计覆盖图形对象所需要的小方块的数目 N 来计算图形的维数，这种方法被称为盒维计数法（Box counting），所得维数被称为盒维 D_b。如果使用不同边长 r 的方块来测量图形对象可以得到不同的 N，即 N 是 r 的函数 $N(r)$。盒维 D_b 的计算式为：

$$D_b = \lim_{r\to 0}\left(\frac{\ln N(r)}{\ln \frac{1}{r}}\right) \tag{21-5}$$

在实际计算中，很难取长度 $r\to 0$ 的比例尺度进行测量。通常的做法是取几个有限长度的比例尺度进行测量得到相应的 $N(r)$ 值，然后利用尺度 r 和 $N(r)$ 的数值绘制双自然对数 $\ln N(r)\sim \ln r$ 图。如果被测对象确实为分形图，那么得到的点将会分布在一条直线上，通过拟合的方法可以计算出相应直线的斜率，这个斜率是一个负数，其绝对值就是盒维 D_b 的近似值。

分形盒维数值的计算程序可以在 MATLAB 网站中查询，相应网址可以在附录 A 中查询。

21.5 小结

本章主要介绍了分形图形的绘制方法。分形图形是一类具有自相似特征的特殊图形。在认识分形图形之前，人们只是对简单的几何图形，如三角形、四边形以及圆形等进行了相应的研究，而对稍微复杂的图形就没有深入的认识了。随着对分形图形的认识，很多图案可以与分形联系起来，从而使人们认识世界的深度和范围进一步扩大。认识分形的第一步就是对绘制分形图形的学习。本章首先介绍一些基本分形图形的绘制，如 Cantor 集、Julia 集、Mandelbrot 集和 Koch 曲线等的画法。然后介绍了利用迭代函数系统绘制分形图形的方法，利用这个方法可以得到不同形式的分形图形，甚至是接近于自然景物的图案。接下来介绍了递归算法绘制分形图形，这个算法需要在绘图之前确定分形图形的自相似性的数学表述，了解不同尺度的单元关系，确定相关的位置与角度表达式，从而可以利用递归关系画图。最后给出了分维的计算方法。

◆ ◆ ◆

第 22 章 元胞自动机

本章包括

- ◆ **奇偶规则** 介绍奇偶规则的定义及相应的程序实现。
- ◆ **砂堆规则** 介绍砂堆规则，同时给出了一个砂漏例子的过程模拟。
- ◆ **细菌生长模型** 介绍细菌生长的模拟，给出元胞自动机实现的例子。
- ◆ **气体扩散** 在 Margolus 近邻上实现了气体扩散过程的模拟。
- ◆ **蚂蚁规则** 结合蚂蚁规则给出了不同初始条件下蚂蚁规则的演化结果。
- ◆ **六边形格子的粒子运动** 给出 6 边形格子上粒子运动的实现，同时给出一个实例。

元胞自动机（Cellular Automata）是一种时间、空间、对象的状态都是离散形式的动力学模型，它是研究非线性科学的重要研究工具。元胞自动机更适合复杂系统时空演化过程的动态模拟。在 1998 年，Chopard 和 Droz 出版《Cellular Automata Modeling of Physical Systems》，对现有元胞自动机理论进行了全面总结。目前这个工具经常出现在一些前沿科学问题研究中，如图像加密、交通问题、人口问题以及一些关于群体性行为的问题。在元胞自动机模型中，对象的状态可以利用矩阵来表示，而矩阵之间的计算正是 MATLAB 的特长。利用 MATLAB 编写元胞自动机程序可以使代码非常简洁，同时矩阵化的代码可以高效地被执行。本章介绍基本元胞自动机模型如何用 MATLAB 语言实现。

22.1 奇偶规则

奇偶规则最早是由 E. Fredkin 提出的，它是一种定义在二维网格上的元胞自动机。每个网格处的状态用矩阵对应点的像素值决定。在经典的模型中，状态一般是两个，因此用 0 和 1 来表示即可。通过设定元胞自动机的规则来控制相邻时刻的状态的变化（如从 S_t 变为 S_{t+1}）。为了使元胞自动机顺利地进行模拟，还需要设定初始条件和边界条件。

奇偶规则下的元胞自动机规则为：

- ◆ 对应于每一个元胞位置 (i,j) 计算出其上、下、左、右 4 个最近邻格点上在 t 时刻的状态值 S_t 以及总和 $M(i,j)$。同时通过下脚标 (i,j) 来遍历整个区域，得到矩阵 M 所有元素的值。
- ◆ 根据 $M(i,j)$ 取值的奇偶性来决定下一时刻 $t+1$ 该点的状态 $S_{t+1}(i,j)$，相应关系可以用下式表示：

$$S_{t+1}(i,j)=\mathrm{mod}[M(i,j),2] \tag{22-1}$$

其中 mod 表示模除函数，即当 $M(i,j)$ 为偶数时，$S_{t+1}(i,j)$ 等于 0；当 $M(i,j)$ 为奇数时，$S_{t+1}(i,j)$ 等于 1。因此这个规则被称为奇偶规则。

对于边界上的点，在考虑区域内的 4 个近邻不可能都选取到，可能取到两个近邻（4 个角点处的元胞）或者 3 个近邻（边界上除角点以外的元胞）。如果不考虑区域以外的元胞，那么相当于把边界以外的元胞（假想存在的）状态值等于 0。此外边界以外假想的元胞还可以存在其他的取值方案，这里假设边界以外的元胞状态值等于 0。

下面介绍四近邻状态值的计算方法。当然可以使用循环来逐点计算近邻的数值，这里给出一种不用循环计算的方法。如果状态矩阵是 $\mathrm{N}\times\mathrm{N}$，那么生成一个 $(\mathrm{N}+2)\times(\mathrm{N}+2)$ 的矩阵 S_m。把矩阵 S_m 中间的 $\mathrm{N}\times\mathrm{N}$ 部分赋值为状态矩阵 S，而矩阵 S_m 周围的一圈元胞可以根据边界条件来赋值。在奇偶规则中，边界设置为全 0 即可。状态矩阵 S_t 上、下、左、右的近邻可以写为：

```
Sm(1:end-2,2:end-1)  % 左近邻
Sm(3:end,2:end-1)    % 右近邻
Sm(2:end-1,1:end-2)  % 上近邻
Sm(2:end-1,3:end)    % 下近邻
```

如此就可以利用上面的表示方法对近邻元胞进行相关的计算了。这里再补充八近邻中的另外 4 个近邻，它们可以表示为：

```
Sm(1:end-2,1:end-2)   % 左上近邻
Sm(3:end,1:end-2)     % 右上近邻
Sm(1:end-2,3:end)     % 左下近邻
Sm(3:end,3:end)       % 右下近邻
```

下面来考虑奇偶规则的 MATLAB 实现。

```
N=256;    % 设置状态矩阵线元胞总数
t1=4;t2=6;
S=zeros(N); % S 是状态矩阵
S(fix(29*N/59):fix(30*N/59),fix(29*N/59):fix(30*N/59))=1;
sum(S(:))
Ii=imshow(1-S,[]);
axis square;      % 设置坐标轴属性
Sm=zeros(N+2); % 生成一个比状态矩阵多 2 行 2 列的矩阵
for k=1:110;
   Sm(2:end-1,2:end-1)=S; % 把 Sm 中间 N×N 的部分赋值为状态矩阵 S

M=Sm(1:end-2,2:end-1)+Sm(3:end,2:end-1)+Sm(2:end-1,1:end-2)+Sm(2:end-1,3:end);
   S=mod(M,2);       % 用模除得到下一时刻状态矩阵的数值
   set(Ii,'CData',1-S); % 更新显示的数据
   pause(0.2);       % 暂停一下显示出动画效果
end
```

上述程序执行过程中，当 t=90 和 t=108 时输出的图形如图 22.1 所示。

随着初始条件不同，得到的演化图案会有很大的差异。很多元胞自动机规则都具有这个特点，从简单的图案生成复杂的图形。对于奇偶规则，这里再举例说明其自复制的特点，即进行 2^n 次迭代。这里初始状态矩阵的形式如图 22.2(a)所示（其为 256×256 像素的矩阵，相应图片文件保存为光盘中\Ch22 文件夹下的 words1.bmp）。利用奇偶规则可以得到如图 22.2 所示的图形（相应的 MATLAB 程序是光盘中\Ch22 文件夹下的 odd_even1.m 文件）。

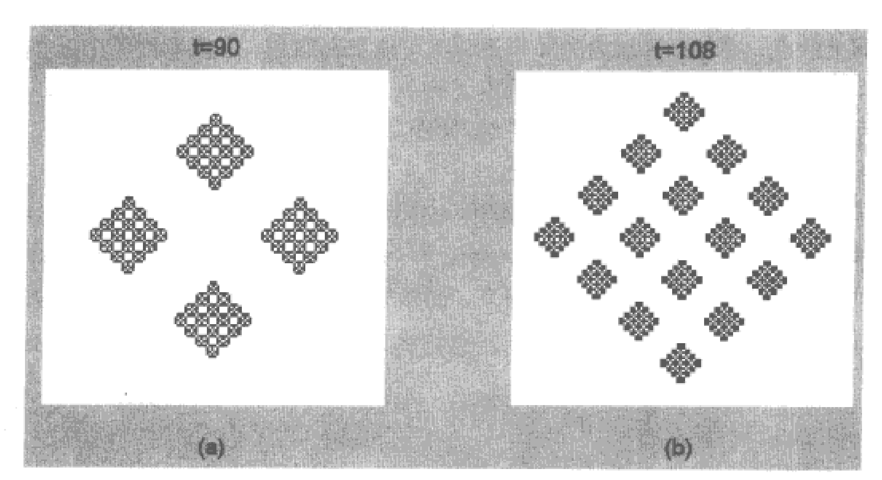


图 22.1 奇偶规则的演示结果

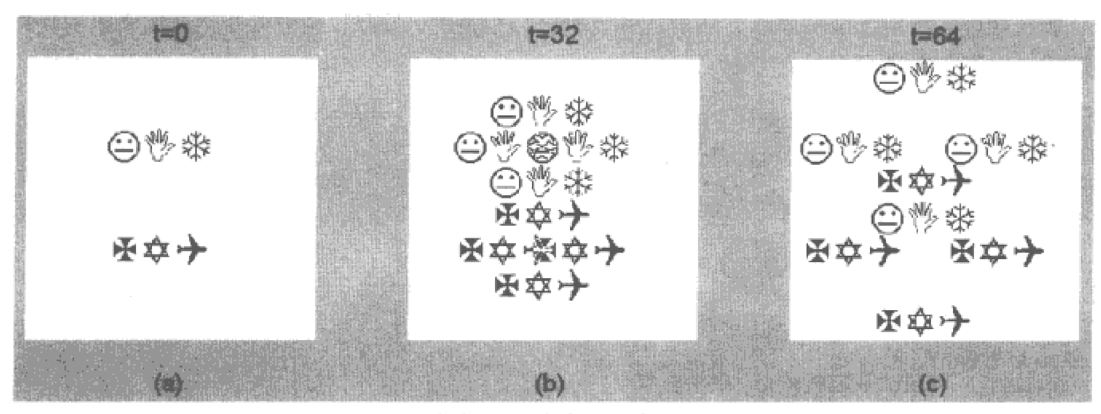


图 22.2 奇偶规则对特殊图案进行演化的结果

从图 22.2 中可以看出出现了初始时的图案。一般在 2^n 次迭代时会出现初始条件描述的图案。图 22.3(a)是时间 $t=128$ 时刻的演化图案，为了对比初始时的图案，把 S_{128} 和 S_0 中的黑色图案相加可以得到如图 22.3(b)所示的结果，可见在图 22.3 中 t=128 初始时的所有元胞状态都已经改变了，这个结论对于其他初始图案也是成立的。

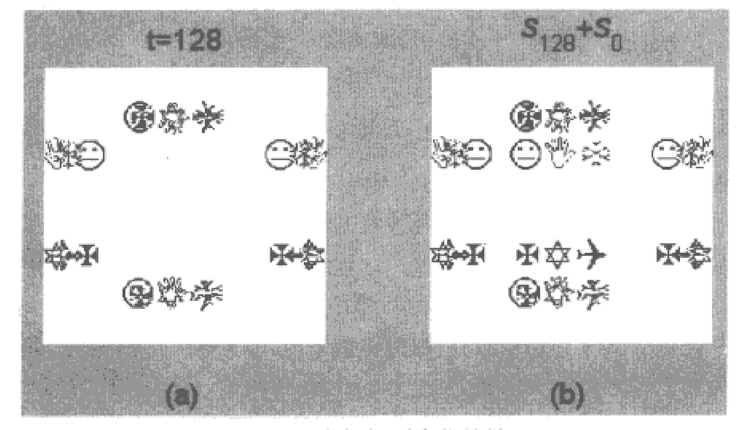


图 22.3 奇偶规则演化的结果

接下来给出元胞自动机常用的两种近邻类型：Von Neumann 型邻居和 Moore 型邻居。前面奇偶规则中的近邻类型就是 Von Neumann 型邻居，即上、下、左、右 4 个最近邻元胞构成的近邻形式。而 Moore 型邻居除了包含最近邻的 4 个元胞外，还有次近邻的 4 个元胞（左上、右上、左下、右下等 4 个位置），所以 Moore 型邻居中有 8 个元胞。另外一种元胞的近邻类型被称为 Margolus 型邻居，其近邻元素是指次近邻的 4 个元胞。

22.2 砂堆规则

颗粒状粒子的物理特性引起了很多研究者的兴趣，这类物质也称为软物质。实际上可以设计一种元胞自动机规则来模拟像砂粒一样颗粒的堆积与倒塌现象。受重力作用，只要下面有空间，砂粒就应该去添补空位。根据这个常识，可以设计相关的元胞自动机规则。

如图 22.4 所示，其中空心圆圈表示砂粒。如果 E 处元胞为空，那么 B 位置的砂粒下落到 E 位置；如果 E 位置有砂粒，那么 B 元胞处的粒子将根据 D 和 F 处粒子是否为空决定下落的方向。

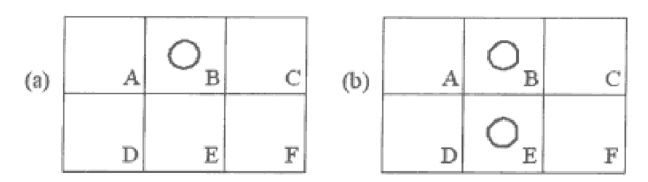


图 22.4 砂堆规则的示意图

同样地，利用数值“1”表示元胞处有砂粒，而用数值“0”表示元胞处无砂粒。如果使用 s_{ul}，s_{ur}，s_{ll} 和 s_{lr} 分别表示时刻 t “左上角”、“右上角”、“左下角”和“右下角”元胞的状态，用 S_{ul}，S_{ur}，S_{ll} 和 S_{lr} 分别表示时刻 $t+1$ “左上角”、“右上角”、“左下角”和“右下角”元胞的状态，那么它们之间的关系可以利用下式表达：

$$S_{ul} = s_{ul}s_{ll}\left[s_{lr} + (1-s_{lr})s_{ur}\right] \quad (22\text{-}2)$$

$$S_{ur} = s_{ur}s_{lr}\left[s_{ll} + (1-s_{ll})s_{ul}\right] \quad (22\text{-}3)$$

$$S_{ll} = s_{ll} + (1-s_{ll})\left[s_{ul} + (1-s_{ur})s_{ur}s_{lr}\right] \quad (22\text{-}4)$$

$$S_{lr} = s_{lr} + (1-s_{lr})\left[s_{ur} + (1-s_{ul})s_{ul}s_{ll}\right] \quad (22\text{-}5)$$

从上面 4 个表达式可以看出，时刻 $t+1$ 砂粒的状态可以使用时刻 t 的砂粒来表示。这几个式子描述了砂粒自由下落的过程，可以认为是元胞自动机的规则。

下面来考虑一个砂漏过程。与自由下落过程不同的是，只有在自由空间内公式（22-2）到（22-4）才是有效的。在砂漏边界外（如图 22.5 左图的深灰色部分）应该约束砂粒不能进入，而且在砂漏最下一层砂粒不能再继续向下运动，因此应该考虑对公式（22-2）到（22-4）进行修正。

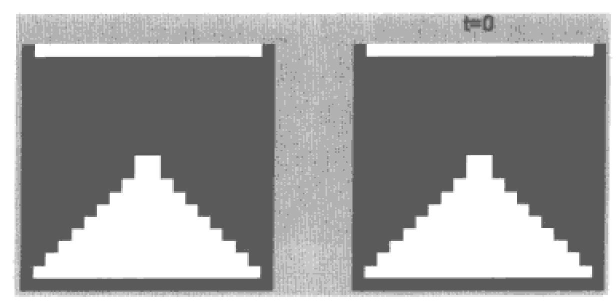


图 22.5　砂漏模型

这里引入边界矩阵 B，边界矩阵 B 某点的元素值如果等于 1 表示砂粒位置不能进入；元素值等于 0 的位置砂粒可以进入。在边界矩阵 B 的限制下，前面的公式（22-2）到（22-5）可以修正为：

$$S_{ul}=B_{ul}s_{ul}+(1-B_{ul})s_{ul}s_{ll}\left[s_{lr}+(1-s_{lr})s_{ur}\right] \tag{22-6}$$

$$S_{ur}=B_{ur}s_{ur}+(1-B_{ur})s_{ur}s_{lr}\left[s_{ll}+(1-s_{ll})s_{ul}\right] \tag{22-7}$$

$$S_{ll}=s_{ll}+(1-s_{ll})\left[s_{ul}(1-B_{ul})+(1-s_{ul})s_{ur}s_{lr}(1-B_{ur})\right] \tag{22-8}$$

$$S_{lr}=s_{lr}+(1-s_{lr})\left[s_{ur}(1-B_{ur})+(1-s_{ur})s_{ul}s_{ll}(1-B_{ul})\right] \tag{22-9}$$

在实际计算中可以使用式（22-6）到（22-9）模拟砂粒下落的过程，计算过程如图 22.6 所示。这里 rn 表示砂漏模型中间漏口列号的位置，它是一个随机数值。从 rn 位置处向左右两侧以 2×2 的单元网格进行计算，而行数 r 从最下面一行逐行计算至上数第二行，这样遍历之后可以得到下一时刻砂粒的位置分布情况。

s_{ul}	s_{ur}	…	s_{ul} (r-1, rn-1)	s_{ur} (r-1, rn)	s_{ul} (r-1, rn+1)	s_{ur} (r-1, rn+2)	…	s_{ul}	s_{ur}
s_{ll}	s_{lr}	…	s_{ll} (r, rn-1)	s_{lr} (r, rn)	s_{ll} (r, rn+1)	s_{lr} (r, rn+2)	…	s_{ll}	s_{lr}

图 22.6　砂漏模型的计算过程

根据上面的分析，可以得到下面的 MATLAB 程序。

```
N=10;          % 控制粒子数的参数
S=zeros(2*N); % 生成状态矩阵 S
S_b=S;         % 设置边界矩阵 S_b
[x,y]=meshgrid(linspace(-N,N,2*N)); % 坐标矩阵
S_b(x-y<1 & x+y<-1 &abs(x)>1)=0.5; % 生成边界矩阵元素值等于 0.5 的地方粒子不能进入
S_b(x-y>1 & x+y>-1 & abs(x)>1)=0.5; % 生成边界矩阵元素值等于 0.5 的地方粒子不能进入
S(x-y>=1 & x+y<=-1)=1; % 布置砂粒的初始分布，元素值等于 1 的位置表示有砂粒
S(1,:)=0;                    % 顶层设置为空
S(end,:)=1;                  % 顶层设置为空
Ss=max(S,S_b); % 联合边界矩阵和状态矩阵，记录边界与砂粒分布的矩阵
B=zeros(2*N); % 标识边界和地面的矩阵 B，B 的数值等于 1 砂粒不能进入，B 的数值等于 0 砂粒能进入
B(x-y<0 & x+y<1 & abs(x)>1)=1; % 设置地面以上的边界矩阵 B 的数值
B(x-y>-1 & x+y>0 & abs(x)>1)=1; % 设置地面以上的边界矩阵 B 的数值
B(end,:)=1;  % 设置地面的矩阵 B 的数值
Ss(end,:)=1; % 把地层设置为 1
```

```
subplot(121);imshow(1-Ss,[]); % 显示初始时图案
subplot(122);Ii=imshow(1-Ss,[]); % 显示初始时图案
ti=title('t=0','Fontsize',14);        % 实时显示时间
for k=1:50;
    for r=2*N:-1:2;
        rn=round(rand);      % 随机选择计算位置
        for r1=N+rn:-1:2;     % 这个循环结构用来计算中心位置向左侧的状态矩阵更替过程
            s_ul=S(r-1,r1-1); % 提取左上角位置的状态矩阵数值
            s_ur=S(r-1,r1);   % 提取右上角位置的状态矩阵数值
            s_ll=S(r,r1-1);    % 提取左下角位置的状态矩阵数值
            s_lr=S(r,r1);      % 提取右下角位置的状态矩阵数值
            B_ul=B(r,r1-1);    % 提取边界状态矩阵像素值（左上角）
            B_ur=B(r,r1);     % 提取边界状态矩阵像素值（右上角）
          S_ul=B_ul*s_ul+(1-B_ul)*s_ul*s_ll*[s_lr+(1-s_lr)*s_ur]; %左上角状态矩阵的
像素值
          S_ur=B_ur*s_ur+(1-B_ur)*s_ur*s_lr*[s_ll+(1-s_ll)*s_ul]; %右上角状态矩阵的
像素值
          S_ll=s_ll+(1-s_ll)*[s_ul*(1-B_ul)+(1-s_ul)*s_ur*s_lr*(1-B_ur)]; %左下角
状态矩阵的像素值
          S_lr=s_lr+(1-s_lr)*[s_ur*(1-B_ur)+(1-s_ur)*s_ul*s_ll*(1-B_ul)]; %右下角
状态矩阵的像素值
            S(r,r1-1)=S_ll;    % 更新状态矩阵左下角位置的数值
            S(r,r1)=S_lr;      % 更新状态矩阵右下角位置的数值
            S(r-1,r1-1)=S_ul;  % 更新状态矩阵左上角位置的数值
            S(r-1,r1)=S_ur;    % 更新状态矩阵右上角位置的数值
        end
        for r2=N+rn+1:2*N-1; % 这个循环结构用来计算中心位置向右侧的状态矩阵更替过程
            s_ul=S(r-1,r2);    % 提取左上角位置的状态矩阵数值
            s_ur=S(r-1,r2+1); % 提取右上角位置的状态矩阵数值
            s_ll=S(r,r2);      % 提取左下角位置的状态矩阵数值
            s_lr=S(r,r2+1);   % 提取右下角位置的状态矩阵数值
            B_ul=B(r,r2);     % 提取边界状态矩阵像素值（左上角）
            B_ur=B(r,r2+1);   % 提取边界状态矩阵像素值（右上角）
          S_ul=B_ul*s_ul+(1-B_ul)*s_ul*s_ll*[s_lr+(1-s_lr)*s_ur]; %左上角状态矩阵的
像素值
          S_ur=B_ur*s_ur+(1-B_ur)*s_ur*s_lr*[s_ll+(1-s_ll)*s_ul]; %右上角状态矩阵的
像素值
          S_ll=s_ll+(1-s_ll)*[s_ul*(1-B_ul)+(1-s_ul)*s_ur*s_lr*(1-B_ur)]; %左下角
状态矩阵的像素值
          S_lr=s_lr+(1-s_lr)*[s_ur*(1-B_ur)+(1-s_ur)*s_ul*s_ll*(1-B_ul)]; %右下角
状态矩阵的像素值
            S(r,r2)=S_ll;       % 更新状态矩阵左下角位置的数值
            S(r,r2+1)=S_lr;     % 更新状态矩阵右下角位置的数值
            S(r-1,r2)=S_ul;     % 更新状态矩阵左上角位置的数值
            S(r-1,r2+1)=S_ur;   % 更新状态矩阵右上角位置的数值
        end
    end
    Ss=max(S,S_b);     % 联合边界矩阵和状态矩阵
    Ss(S_b>0.2)=0.5;    % 把地面以上的边界部分灰度值设置为 0.5
    set(Ii,'CData',1-Ss);   % 更新联合状态矩阵 Ss
    set(ti,'String',['t = ',num2str(k)]); % 更新时间
    pause(0.2);    % 暂停一下以显示动画效果
end
```

首先设置砂漏的边界，利用矩阵 S_b 表示，同时初始化砂粒所对应的状态矩阵 B。然后根据

规则计算不同时刻的状态矩阵演化情况。这里把状态矩阵和边界矩阵结合在一起实时显示。上述程序所得结果如图 22.7 所示，从中可以看出砂粒下落的情况。

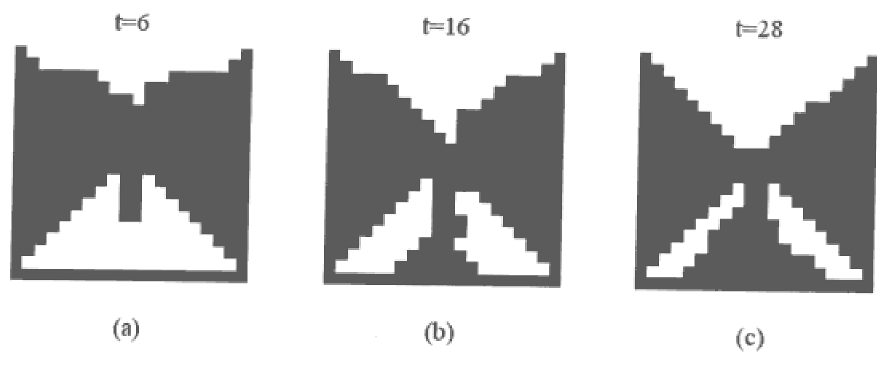


图 22.7 砂漏模型的截图

可以适当地增加参数 N 的取值（如 100），同时时间模拟的步数增加至可以模拟更为精细的界面。当然这个砂漏过程的模拟只是一个粗糙的过程，其中未考虑砂粒之间的摩擦以及重力因素。

22.3 细菌生长模型

本节来研究在二维平面上细菌繁殖的过程，其基本模型是某元胞处的细菌选择八近邻中的空位向外生长。当存在多个空位置时，随机选择一个空位。同时对细菌做如下限定：细菌的寿命为 τ，即当某处细菌的生命大于 τ 的时候细菌死亡，该位置变为空位；细菌存在一定的死亡率 p，即每个时间步内细菌受一些外界因素的影响会死亡；在细菌的生命力大于 τ_0 的时候才能开始繁殖，之前处于幼年期。边界的条件为封闭性，即边界以外的位置细菌不能生长。

根据描述可以得出如图 22.8 所示的流程图。

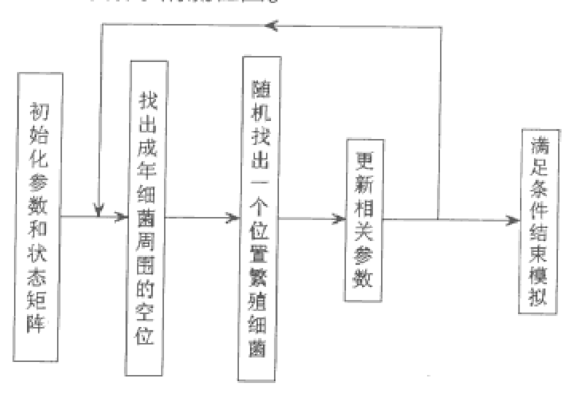


图 22.8 细菌生成模拟的流程图

利用二值矩阵表示是否存在细菌，其中数值 1 表示存在细菌，0 表示空位。使用矩阵 L 来记录细菌的年龄。这里细菌寿命为 8，死亡率为 0.05。相关的 MATLAB 程序如下：

```
tau=8;  % 定义细菌的寿命
p=0.05; % 定义细菌的死亡率
tau0=2; % 定义细菌的繁殖年龄
N=200; % 定义方形状态矩阵的行数
```

```
S=zeros(N); % 生成状态矩阵 S
S(N/2:N/2+1,N/2:N/2+1)=1; % 初始化状态矩阵
L=3*S;     % 初始化生命值
Ii=imshow(1-S); % 显示状态矩阵
dx=[-1,-1,-1,0,0,1,1,1];    % 用相对位移来生成近邻坐标，dx 用于生成行位置
dy=[-1,0,1,-1,1,-1,0,1];    % 用相对位移来生成近邻坐标，dy 用于生成列位置
ti=title('t=0','Fontsize',14);  % 实时显示时间
for k=1:300;
    [x,y]=find(S>0.5 & L>tau0+0.5); % 找出成年细菌
    Po=zeros(N); % 临时矩阵，用于标记找出的空位
    for n=1:length(x);
        xt=x(n);  % 提取成年细胞的位置(行位置)
        yt=y(n);  % 提取成年细胞的位置(列位置)
        xt=xt+dx; % 计算八近邻的位置(行位置)
        yt=yt+dy; % 计算八近邻的位置(列位置)
        xt(yt<0.5|yt>N+0.5)=[]; % 除去边界以外的近邻
        yt(yt<0.5|yt>N+0.5)=[]; % 除去边界以外的近邻
        yt(xt<0.5|xt>N+0.5)=[]; % 除去边界以外的近邻
        xt(xt<0.5|xt>N+0.5)=[]; % 除去边界以外的近邻
        Ind=sub2ind([N,N],xt,yt); % 转化为索引坐标
        cc=find(S(Ind)<0.5 & Po(Ind)<0.5);  % 找出近邻中的空位
        ra=randperm(length(cc));   % 产生一个随机序列
        ra(2:end)=[];        % 删去第二个以后的元素
        Po(xt(ra),yt(ra))=1;  % 标记空的位置
    end
    S(Po>0.5)=1;  % 把选出的空位设置为细菌
    L=L+[S>0.5];  % 生命值累加
    R=rand(N);   % 生成随机矩阵，控制细菌的死亡
    S(R<p)=0;   % 死亡细菌出的细菌状态置 0
    L(R<p)=0;   % 死亡细菌出的生命值置 0
    S(L>tau+0.5)=0;  % 到达寿命时细菌死亡
    L(L>tau+0.5)=0;  % 到达寿命时，细菌生命力置 0
    set(Ii,'CData',1-S); % 更新显示矩阵
    set(ti,'String',['t = ',num2str(k)]); % 更新显示时间
    pause(0.2); % 暂停一下以显示动画效果
end
```

执行上面的程序可以模拟细菌的生长过程。下面给出时间在$t=100$，$t=200$和$t=300$时的图案，如图 22.9 所示。

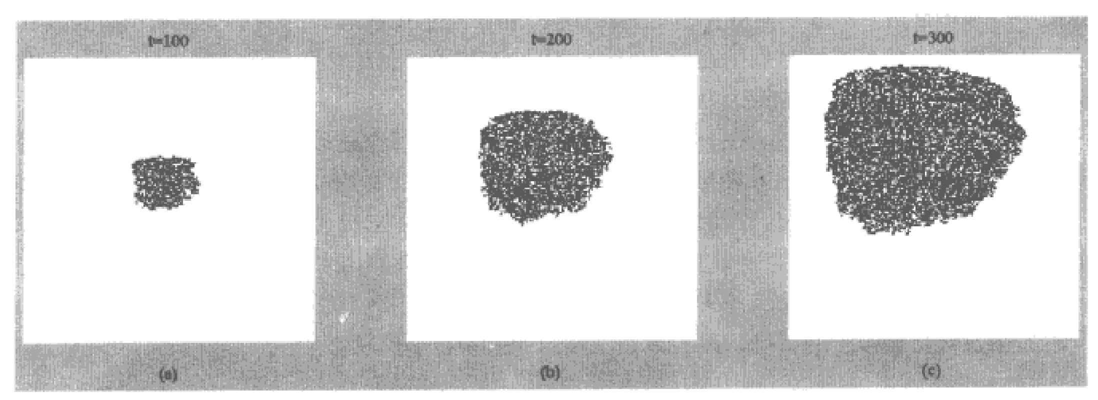


图 22.9　细菌生成模拟得到的图案

把语句“Sc(k)=sum(sum(L==3))/sum(S(:));”放在语句“pause(0.2);”的下一行，可以用来计算年龄为 3 的细菌占细菌的比例。同时利用下面的语句绘制相应的曲线：

```
figure;plot(Sc); % 绘图
xlabel('Time','Fontsize',14,'Fontname','Times new roman'); % X 轴标注
ylabel('{\itL}_3/{\itN}_{total}','Fontsize',14,'Fontname','Times new roman'); %
Y 轴标注
```

进行上面补充之后执行程序可以得到如图 22.10 所示的曲线，即年龄为 3 的细菌占细菌总数随时间变化的曲线。可以看出在最初时这个比例值起伏比较大，在 $t=200$ 以后这个比例只是在小的范围内变化，达到一个平衡状态。类似地，可以计算其他相关的曲线来反映细菌生成过程的特征。

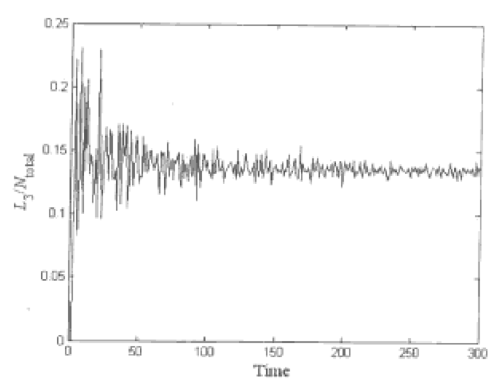


图 22.10　年龄为 3 的细菌占总细菌数的比例数值曲线

可以变化相关的参数，如细菌寿命、死亡率等以研究细菌的生长问题。此外还可以对生长模型进行修正以深入研究。在这个细菌生长过程模拟中，使用随机数来控制细菌的死亡，以及八近邻空位中的随机选择。而且与前面不同的是，除了元胞状态矩阵之外还存在着其他中间控制矩阵，如年龄矩阵 L、标记空位情况的矩阵 Po 等。

22.4　气体扩散

HPP 规则最早是由法国科学家 Hardy，Pomeau 和 Pazzis 提出的，因此其名称由这 3 个人名字的首字母缩写而成。它是一种重要的元胞自动机，用于描述离散形式的格子气自动机时空演变情况。其基本研究对象是按适当规则在元胞位置上运动的粒子。元胞状态是二值的，即“1”表示该位置存在粒子，“0”表示该位置为空。

下面考虑粒子的运动规则。在一个 2×2 区域上研究粒子状态的改变。如图 22.11 所示（灰点表示该位置有粒子），其中给出 6 种可能状态的变化情况，实际上共有 16 种可能的状态。在图 22.11 中，图(a)和图(f)没有发生变化，图(b)、图(c)和图(e)的规律可以总结为：$t+1$ 时刻粒子的状态可以表示为前一时刻 t 的粒子沿主、副对角线方向交换粒子状态，而图(a)和图(f)也可以认为是这样规则得到的结果。但是图(d)的变化结果对应着粒子之间的非平凡碰撞，粒子的状态变化不在对角线上，而是沿上下方向交换粒子的状态，或者说沿左右方向交换粒子的状态。

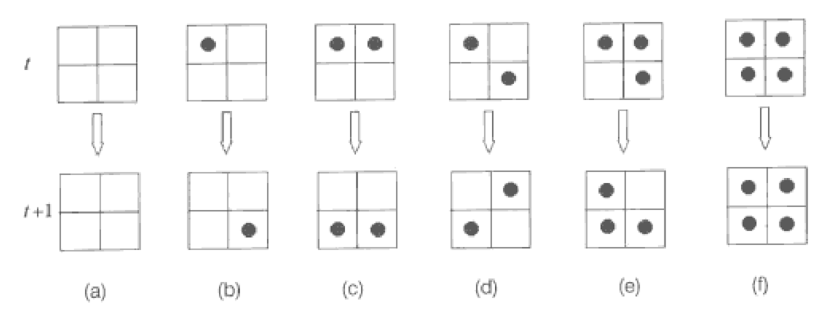


图 22.11 不同情况粒子状态变化的规律

为了使粒子演化具有流动性，这里在时间步为奇数时选择点划线围成的区域计算，而在偶数时间步内选择虚线标记的区域进行计算。也就是说在偶数时间步内，计算区域是从区域左上角开始计算，而奇数时间步内将分别在水平和竖直方向上向和右向下移动一个元胞格位。

如图 22.12 所示，一个矩形封闭区域内被一个带空的隔板分为两个部分。在计算中把左侧的矩形区域视为“一区”，中间隔板上的空对应的狭窄区域视为“二区”，右侧矩形区域视为“三区”。区域处的衔接方式要和图 22.13 所示的时间步选择相匹配，从而“二区”部分可以成为粒子在“一区”和“三区”之间过往的通道。在计算中要限定粒子只能在可能进入的元胞位置上运动。在初始时粒子的布局如图 22.14 所示。

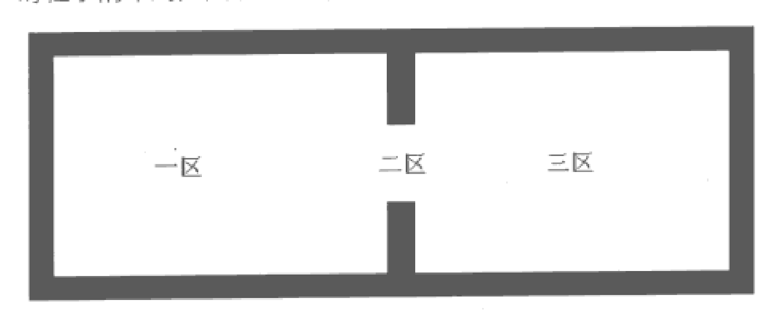


图22.12 计算的分区方式

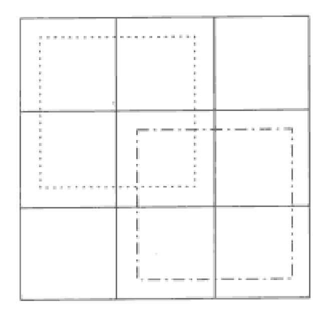


图22.13 按时间步选择的计算区域

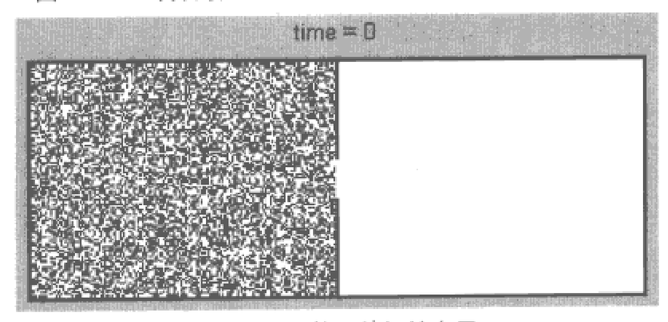


图 22.14 粒子的初始布局

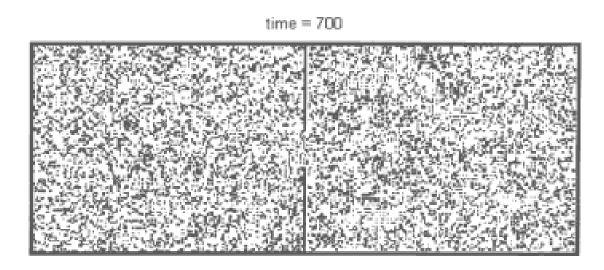
根据上面的分析，可以编写出相应的实现程序，内容如下：

```
function diffuse
rand('state',0);
set(gcf,'DoubleBuffer','on');
N=300; % 设置状态矩阵的列数
M=120; % 设置状态矩阵的行数
```

```
h=2; % 设置边界的宽度
xp=round(N/2); % 计算中间隔板的位置
D=round(M/12); % 计算中间开孔的半宽度
B=ones(M,N); % 生成边界矩阵 B，其元素等于 0 的位置粒子不能进入，等于 1 的位置可以进入
B(1:h,:)=0; % 设置上侧边界
B(:,1:h)=0; % 设置左侧边界
B(:,N-h+1:N)=0; % 设置右侧边界
B(M-h+1:M,:)=0; % 设置下面的边界
B(:,xp:xp+h-1)=0; % 设置中间的隔板
B(round(M/2)-D+1:round(M/2)+D,xp:xp+h-1)=1; %  在隔板中间开孔
S=zeros(M,N); % 生成粒子状态矩阵，等于 1 的位置表示有粒子，等于 0 的位置表示没有粒子
S(h+1:M-h,h+1:xp-1)=[rand(M-2*h,xp-h-1)>0.5]; % 在左侧区域上随机布上粒子
Tc=700; % 设置演化过程的总时间
C=min(1-S,B); % 综合状态矩阵与边界矩阵 B 为显示矩阵 C
ax=subplot(211);asa=imshow(C,[]); % 显示初始时的粒子分布图案
ti=title('time = 0'); % 在图题出添加时间值
rL=sum(sum(S(:,1:xp)))/[(xp-h)*(M-2*h)]; % 计算左侧粒子密度
rR=sum(sum(S(:,xp+h:N)))/[(N-xp-2*h+1)*(M-2*h)]; % 计算右侧粒子密度
subplot(212);p1=plot(1,rL,'r-');xlim([1,Tc]);% 绘制左侧密度曲线
hold on;p2=plot(1,rR,'k--');% 绘制右侧密度曲线
xlabel('时间'); % X 轴标注
ylabel('时间'); % Y 轴标注
legend('左侧密度','右侧密度'); % 加注图例
for k=1:Tc;
    p=mod(k,2); % 判断奇数或者偶数时间步
    x1=h+1+p:2:xp-3+p; % 计算一区位置坐标数据（X 轴）
    y1=h+1+p:2:M-h-2+p; % 计算一区位置坐标数据（Y 轴）
    x2=xp-1+p:2:xp+h-1+p; % 计算二区位置坐标数据（X 轴）
    y2=round(M/2)-D+2:2:round(M/2)+D-1; % 计算二区位置坐标数据（Y 轴）
    x3=xp+h+1+p:2:N-h-2+p; % 计算三区位置坐标数据（Y 轴）
    y3=h+1+p:2:M-h-2+p; % 计算三区位置坐标数据（X 轴）
    St=S; % 初始化中间矩阵 St
    St(y1,x1)=S(y1+1,x1+1); % 一区交叉换位
    St(y1+1,x1+1)=S(y1,x1); % 一区交叉换位
    St(y1,x1+1)=S(y1+1,x1); % 一区交叉换位
    St(y1+1,x1)=S(y1,x1+1); % 一区交叉换位
    St(y2,x2)=S(y2+1,x2+1); % 二区交叉换位
    St(y2+1,x2+1)=S(y2,x2); % 二区交叉换位
    St(y2+1,x2)=S(y2,x2+1); % 二区交叉换位
    St(y2,x2+1)=S(y2+1,x2); % 二区交叉换位
    St(y3,x3)=S(y3+1,x3+1); % 三区交叉换位
    St(y3+1,x3+1)=S(y3,x3); % 三区交叉换位
    St(y3+1,x3)=S(y3,x3+1); % 三区交叉换位
    St(y3,x3+1)=S(y3+1,x3); % 三区交叉换位
    St=changep(St,S,x1,y1); % 计算一区非平凡碰撞
    St=changep(St,S,x2,y2); % 计算二区非平凡碰撞
    St=changep(St,S,x3,y3); % 计算三区非平凡碰撞
    S=St; % 更新状态矩阵 S 的数值
    set(ti,'String',['time = ',num2str(k)]); % 更新图题处的时间数值
    set(asa,'CData',min(1-S,B)); % 更新粒子状态的图案
    rL(k+1)=sum(sum(S(:,1:xp)))/[(xp-h)*(M-2*h)]; % 计算左侧粒子密度
    rR(k+1)=sum(sum(S(:,xp+h:N)))/[(N-xp-2*h+1)*(M-2*h)]; % 计算右侧粒子密度
    set(p1,'XData',1:k+1,'YData',rL); % 更新左侧密度曲线的数据
    set(p2,'XData',1:k+1,'YData',rR); % 更新右侧密度曲线的数据
    pause(0.2); % 暂停一下，显示动画效果
end
```

```
function St=changep(St,S,x,y);
% 这里使用水平方向和竖直方向同时求和，当所有和值都等于 1 时即为对角形式
Sr1=S(y+1,x)+S(y,x); % 第一列竖直方向求和
Sc1=S(y,x+1)+S(y+1,x+1); % 第二列竖直方向求和
Sr2=S(y,x+1)+S(y,x); % 第一行水平方向求和
Sc2=S(y+1,x)+S(y+1,x+1); % 第二行水平方向求和
[p,q]=find(Sr1==1&Sc1==1&Sc2==1&Sc2==1); % 找出 4 个和值都等于 1 的位置
for k=1:length(p);
    St(y(p(k)),x(q(k)))=S(y(p(k))+1,x(q(k))); % 第一列上下交换赋值
    St(y(p(k))+1,x(q(k)))=S(y(p(k)),x(q(k))); % 第一列上下交换赋值
    St(y(p(k)),x(q(k))+1)=S(y(p(k))+1,x(q(k))+1);  % 第二列上下交换赋值
    St(y(p(k))+1,x(q(k))+1)=S(y(p(k)),x(q(k))+1); % 第二列上下交换赋值
End
```

首先生成粒子的初始布局和边界情况，然后根据粒子扩散的规律计算不同时刻粒子位置变化情况，并把粒子各时刻位置以动画形式显示出来。其中子函数 changep 用于计算粒子位置的变化。上述程序是一个函数文件的格式，把上面的程序保存为 diffuse 即可执行并得到相应的模拟结果。执行上述程序结束后可以得到如图 22.15 所示的结果。



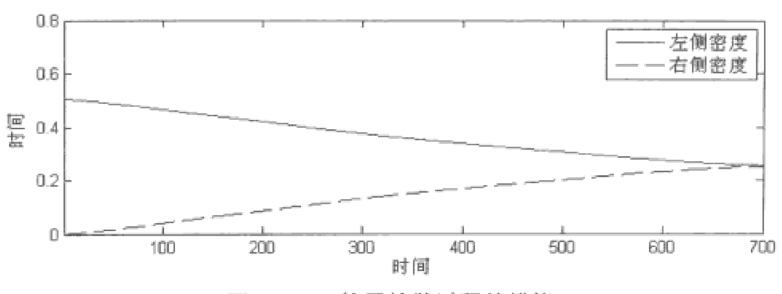


图 22.15 粒子扩散过程的模拟



在模拟粒子向右侧空区域扩散的过程中，实时地显示了左右两侧粒子密度的变化情况。可见在一定时间后两侧粒子密度相等，达到了平衡状态。

在下面的网址中可以下载到格子气模拟的小程序：http://instruct1.cit.cornell.edu/courses/bionb441/CA/gas2.m。感兴趣的读者可以研究其实现的算法。

22.5 蚂蚁规则

蚂蚁规则是由 C. Langton 和 G. Turk 提出的一种元胞自动机，这个规则通过简单的运动算法来

模拟蚂蚁在地面上的行为。在一个二维区域内，元胞的状态用黑白两种颜色表示，其中黑色对应矩阵元素的数值为 0，白色对应矩阵元素的数值等于 1。蚂蚁运动到白色的元胞时，对应的元胞变为黑色同时蚂蚁的运动方向逆时针转 90°；蚂蚁运动到黑色的元胞时，对应的元胞变为白色同时蚂蚁的运动方向顺时针转 90° 。上面就是蚂蚁规则的内容，可见这个规则非常简单，但它却是一个复杂的演化过程。

下面考虑蚂蚁规则的程序实现。这里设置蚂蚁初始的位置是(x_0, y_0)，蚂蚁在每个时间间隔内移动的长度是 1。这里把蚂蚁的位置利用一个复数表示，即：

$$z = x_0 + y_0 \mathrm{i} \tag{22-10}$$

利用一个参数 A 来控制蚂蚁运动方向的改变。这里 A 是一个在单位圆上的复数，其可能取值为$\{-1,1,-\mathrm{i},\mathrm{i}\}$ 4 个数中的一个，其中-1 和 1 分别表示向左和向右运动，-i 和 i 分别表示向下和向上运动。蚂蚁运动方向的改变可以使用下面的表达式来实现。

$$A = A(-1)^{1+S(x,y)}\mathrm{i} = \begin{cases} (-\mathrm{i})A, & S(x,y)=0 \\ \mathrm{i}A, & S(x,y)=1 \end{cases} \tag{22-11}$$

其中$S(x, y)$是状态矩阵的数值，式（22-11）可以实现在黑色元胞处顺时针旋转 90° 、在白色元胞处逆时针旋转 90° 。根据上面的分析，可以设计出蚂蚁规则实现的流程图，如图 22.16 所示。

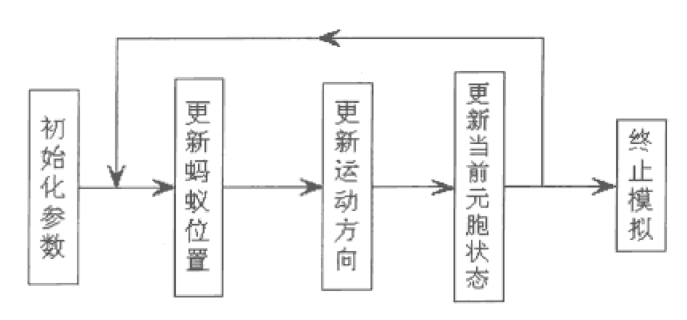


图 22.16　蚂蚁规则实现的流程图

这里取参数如下：状态矩阵的大小是 256×256，一只蚂蚁最初处于(128, 128)位置处，该处元胞的颜色为黑色，其他点的颜色为白色，初始时蚂蚁向左运动（即 A=i），模拟的时间长度是 12000 步。在流程图的基础上可以得到蚂蚁规则的实现程序，具体内容如下：

```
N=256;      % 设置方形状态矩阵的大小
S=ones(N); % 设置方形状态矩阵为全 1 矩阵
x=fix(N/2);  % 计算初始时蚂蚁的位置
y=fix(N/2);  % 计算初始时蚂蚁的位置
S(x,y)=0;    % 设置初始时蚂蚁所在位置为黑色
z=x+y*i;     % 用一个复数表示蚂蚁的位置
Ii=imshow(S,[]); % 显示状态矩阵
kk=12000; % 设置总的时间步数
A=i; % 设置初始时蚂蚁运动的方向
ti=title('time = 0','Fontsize',14,'Fontname','Times New Roman'); % 实时显示时间
for k=1:kk;
```

```
    z=z+A; % 更新蚂蚁的位置，沿运动方向前移一个距离
    x=real(z); % 从复数坐标中取位置坐标
    y=imag(z); % 从复数坐标中取位置坐标
    A=A*i*(-1)^(1+S(x,y)); % 根据当前元胞位置的数值更新蚂蚁的运动方向
    S(x,y)=~S(x,y); % 改变当前位置的颜色
    set(Ii,'CData',S); % 更新显示的状态矩阵
    set(ti,'String',['time = ',num2str(k)]); % 更新显示时间
    pause(0.1); % 暂停一下显示动画效果
end
```

执行上面的程序可以实现蚂蚁规则，在 t=2000, 4000, 6000, 7000, 8000, 9000, 10000, 11000 和 12000 等时间点处的图案如图 22.17 所示。可见在一定时间之后蚂蚁将会离开混沌的状态，并沿左上角 45° 方向运动。

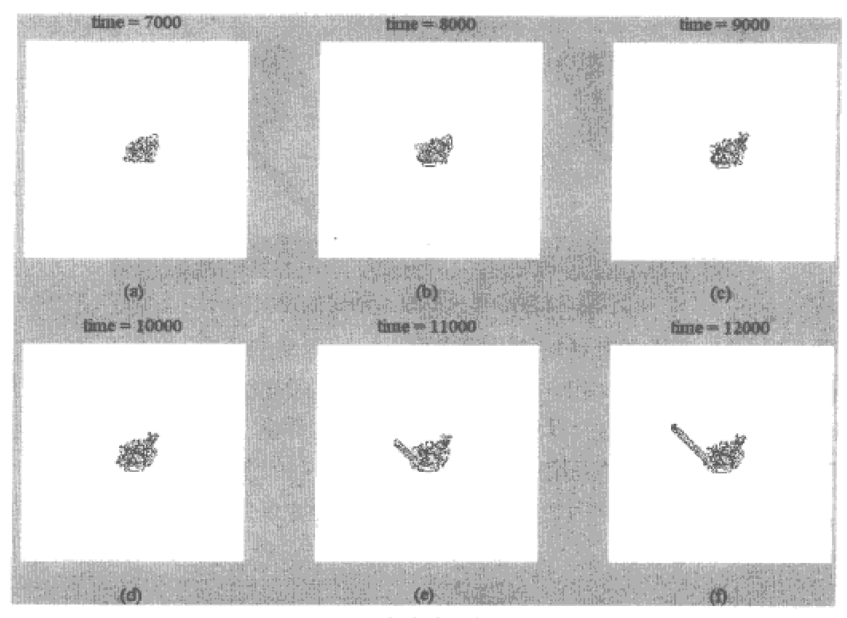


图 22.17 蚂蚁规则得到图案

前面研究了一只蚂蚁在平面上运动的情况，下面来考虑多只蚂蚁在平面上运动的情况。这里给出 4 只蚂蚁在平面上运动的情况，把 4 只蚂蚁布局在状态矩阵的中心处相邻的 4 个位置。它们具有不同的初始速度，如图 22.18 所示。其中的箭头表示蚂蚁运动的初始方向，A，B，C 和 D 处元胞初始时的颜色设置为黑色，状态矩阵其他位置的元胞设置为白色。在每个时间间隔内，执行蚂蚁规则按初始时 A，B，C 和 D 4 只蚂蚁的先后顺序进行计算。根据前面介绍的蚂蚁规则，实现多只蚂蚁的运动行为的研究。程序实现的结构如图 22.16 所示。MATLAB 程序内容可以参见光盘中\Ch22 文件夹下的 langton4.m 文件。

在图 22.18(a)所示的初始条件下，选择状态矩阵的大小为 200×200。运行 2000 步可以得到如图 22.19 所示的图形。可见在这样的情况下，4 只蚂蚁没有在中心区域进行往复的混乱运动，而是直接进入分离的状态且它们都是在与状态矩阵主、副对角线平行的方向上运动。

下面再考虑在图 22.18(b)所示的初始条件下的演化结果。取状态矩阵的大小为 512×512，模拟时间总长度是 23000，在时间 $t=13000$ 和 14000 时的图案如图 22.20 所示。

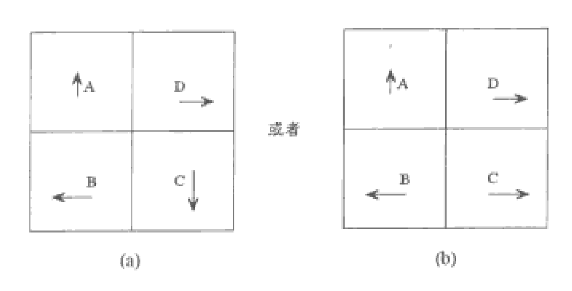


图 22.18　4 只蚂蚁的初始状态示意图

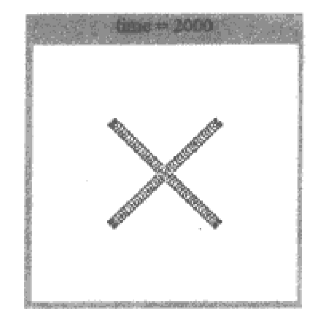


图 22.19　4 只蚂蚁爬行而得到的图案

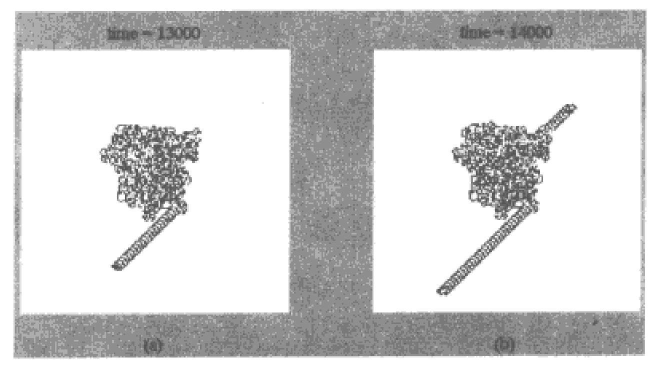


图 22.20　4 只蚂蚁运动时的图案

可以选择更大的状态矩阵或者更长的时间来模拟蚂蚁运动的情况，此外不同的初始条件对结果影响很大。下面再来考虑一种状态矩阵初始时不同的情况，即在状态矩阵中有一条黑色的元胞状态来研究蚂蚁穿过黑色条的情况，相应程序在光盘中\Ch22 文件夹下的 langton1w.m 文件中。执行这个程序可以得到如图 22.21 所示的图案，其中给出了不同时刻的演化结果。从图 22.21 中可以看出，在蚂蚁未到达黑色条时与前面图 22.17 所示的行为一致。但是蚂蚁运动到黑色条时，其将围绕黑色条运动，如图 22.21(f)所示。

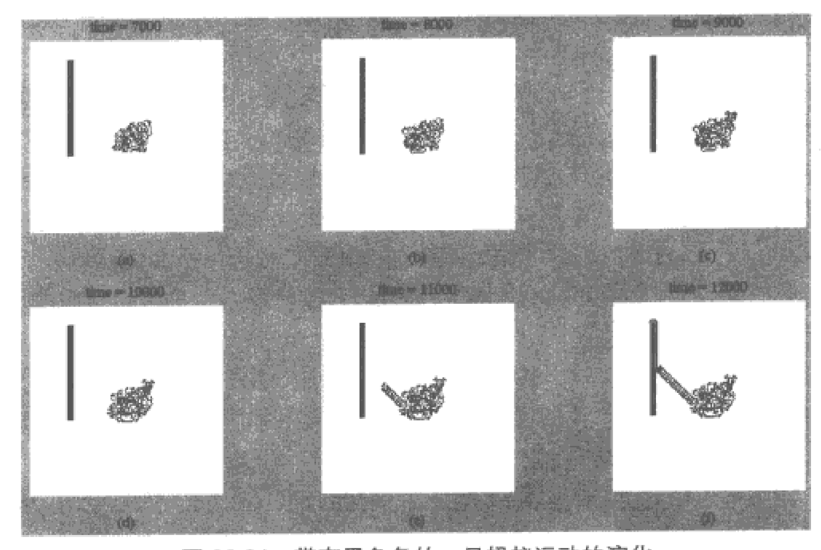


图 22.21　带有黑色条的一只蚂蚁运动的演化

通过本节介绍的蚂蚁规则可以看出一个简单的元胞自动机模型，设置不同的初始条件可以演化出非常丰富的图案。感兴趣的读者可以在前面介绍的基础上研究其他情况的结果。

22.6 六边形格子的粒子运动

前面介绍的元胞自动机的状态矩阵描述的都是矩阵，它所能表达的元胞是一系列的方形格子。研究的粒子只能在水平和竖直两个方向匀速率运动。本节来研究在六边形格子上的粒子运动行为。这种元胞自动机的格子状态如图 22.22 所示，其中粒子可以在 6 个不同的方向上同速率运动。在图 22.22 中，不能直接使用矩阵来表述其中的格子位置，但可以把总的网格分为两类，即白色和灰色两种网格。这时白色网格和灰色网格都是正交的阵列，因此可以使用两个矩阵 W 和 G，其中 W 表示白色圆圈对应的格子位置的状态值，G 表示灰色圆圈对应的格子位置的状态值。元胞自动机的规则就是在两个状态矩阵之间交替地作用。

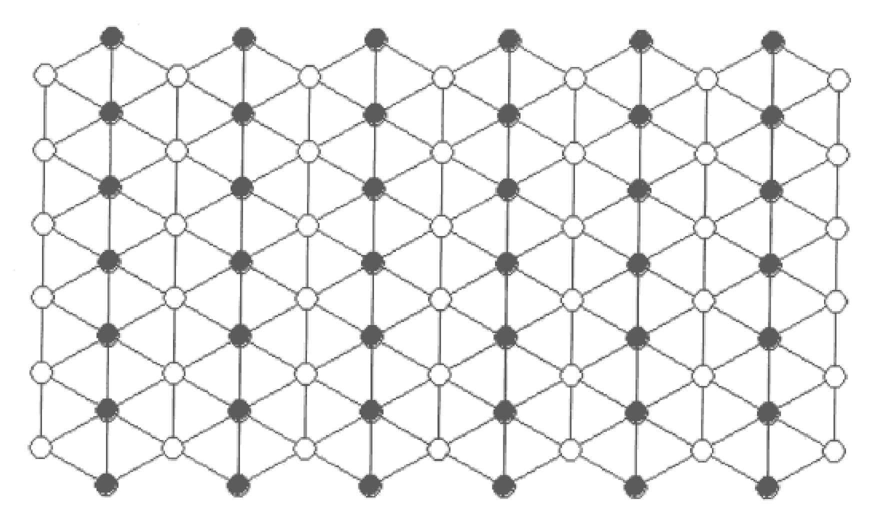


图 22.22 六边形格子的片断

下面给出一个六边形格子上元胞自动机的例子。这里给出一个容器内粒子受重力作用下落的过程模拟。如图 22.23 所示，在 t 时刻，A 处有一个粒子（其被网格标注），而 B 处和 D 处的格位是空的，那么 A 处的粒子将有可能运动到 B 处或者 D 处。至于运动到 B 处还是 D 处，要根据 A 处粒子受到上面粒子的作用力来决定：如果 A 受到的合力分量向左，那么 A 处粒子将会运动到 B 处，反之则运动到 D 处；当受力左右平衡时，那么 A 处的粒子将会随机地运动到 B 处或者 D 处，这里使用概率 p 来控制。对于 B 处或者 D 处只有一处为空时，A 处粒子将运动到空缺的格位。如果 B 处和 D 处都有粒子占据的时候，A 处的粒子将停留在原处。

如图 22.24 所示，是研究其中黑色区域粒子受重力作用下落的过程模拟。粒子被束缚在一个圆形的回转体内，粒子从下面的环形开口下落，左图显示的是一个过中心轴的截面，右图是顶层粒子形成的曲线。主程序如下：

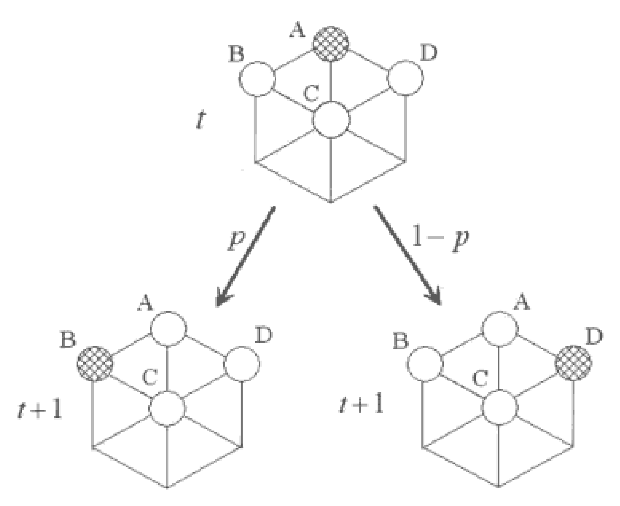


图 22.23　六边形格子的演化规则示意图

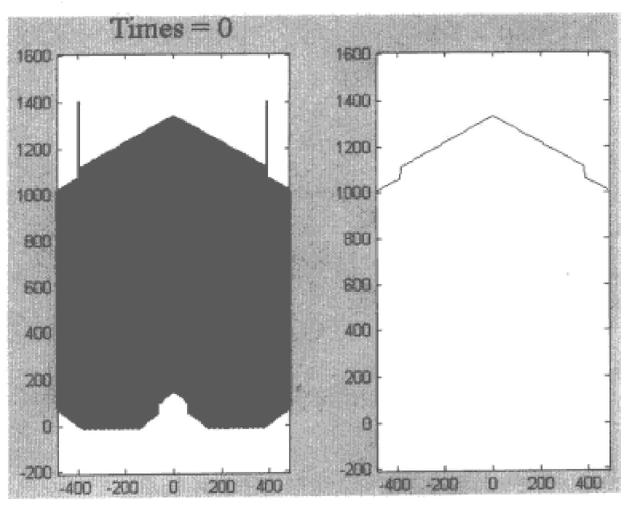


图 22.24　回转体内粒子下落的模拟

```
Ny=116;                  % 线粒子总数
Nx=round(Ny/[Ny/[109/4.47]]);
A0=pi/3;    % 元胞格子的角度
dx=20;      %格子在 X 轴方向的边长
dy=dx/tan(A0); %格子在 Y 轴方向的边长
[Xb,Yb]=meshgrid([0:Nx]*dx,dy/2+[0:Ny-1]*dy);%蓝色粒子单侧坐标
[Xr,Yr]=meshgrid([0:Nx-1]*dx+dx/2,[0:Ny]*dy);%红色粒子单侧坐标
Xb=[-fliplr(Xb(:,2:end)),Xb];     % 关于 x=0 轴对称处理
Yb=[fliplr(Yb(:,2:end)),Yb];      % 关于 y=0 轴对称处理
Xr=[-fliplr(Xr),Xr];              % 关于 x=0 轴对称处理
Yr=[fliplr(Yr),Yr];               % 关于 y=0 轴对称处理
Mr=ones(size(Xr));              % 红色粒子状态矩阵
Mb=ones(size(Xb));             % 蓝色粒子状态矩阵
R=max(Xb(:));                 % 炉膛的半径
H=max(Yr(:));                % 炉膛的高度
x1=8/10;Kc=sqrt(1.5);
Mb(Xb-Kc*Yb>R*x1)=-1;        % 切去边界下角
```

```
Mr(Xr-Kc*Yr>R*x1)=-1;      % 切去边界下角
Mb(Xb+Kc*Yb<-R*x1)=-1;   % 切去边界下角
Mr(Xr+Kc*Yr<-R*x1)=-1;     % 切去边界下角
x2=0.3;d2=0.3;                                           % 切分流器
Mb(abs(Xb)<R*x2&Xb+Kc*Yb<R*d2&Xb-Kc*Yb>-R*d2)=-1;    % 切分流器
Mr(abs(Xr)<R*x2&Xr+Kc*Yr<R*d2&Xr-Kc*Yr>-R*d2)=-1;       % 切分流器
x3=0.4;Ka=0.5;                                           % 切分流器
Mb(abs(Xb)<R*x2*Ka&Xb+Kc*Yb<R*x3&Xb-Kc*Yb>-R*x3)=-1;  % 切分流器
Mr(abs(Xr)<R*x2*Ka&Xr+Kc*Yr<R*x3&Xr-Kc*Yr>-R*x3)=-1;     % 切分流器
x4=0.8;
y1=1.67;
Mb(abs(Xb)>R*x4&[Xb+sqrt(3)*Yb>H*y1|Xb-sqrt(3)*Yb<-H*y1])=-1;% 切去边界上部
Mr(abs(Xr)>R*x4&[Xr+sqrt(3)*Yr>H*y1|Xr-sqrt(3)*Yr<-H*y1])=-1;    % 切去边界上部
D1=1.73;
Mb(abs(Xb)<=R*x4&[Xb+sqrt(3)*Yb>H*D1|Xb-sqrt(3)*Yb<-H*D1])=-1;% 切去边界上部
Mr(abs(Xr)<=R*x4&[Xr+sqrt(3)*Yr>H*D1|Xr-sqrt(3)*Yr<-H*D1])=-1;    % 切去边界上部
subplot(121)
P1=plot(Xb(Mb==1),Yb(Mb==1),'ro','markerfacecolor','r','markersize',4);
hold on;
P2=plot(Xr(Mr==1),Yr(Mr==1),'ro','markerfacecolor','r','markersize',4);
ks=0.5;
plot([R*x1+dx*ks,R+dx*ks-dx/4,R+dx*ks-dx/4,R*x4+dx*ks,R*x4+dx*ks],...
    [0,(1-x1)*R/Kc,H*0.76,H*0.76+(1-x4)/sqrt(3)*R,H+3*dy+30],...
    'k','linewidth',2);   % 画右边界
plot(-[R*x1+dx*ks,R+dx*ks-dx/4,R+dx*ks-dx/4,R*x4+dx*ks,R*x4+dx*ks],...
    [0,(1-x1)*R/Kc,H*0.76,H*0.76+(1-x4)/sqrt(3)*R,H+3*dy+30],...
    'k','linewidth',2);   % 画左边界
ks=0.4;
plot([-R*x2+dx*ks,-R*x2*Ka+dx*ks,-R*x2*Ka+dx*ks,0,R*x2*Ka-dx*ks,R*x2*Ka-dx*ks,
R*x2-dx*ks],...
    [0,R*x2/Kc/2.3,R*x2/Kc*0.9,R*x3/Kc-dx/2/Kc,...
       R*x2/Kc*0.9,R*x2/Kc/2.3,0],...
    'k','linewidth',2);  % 画分流器
set(gcf,'DoubleBuffer','on'); % 控制动画效果的闪动
xlim([-486,486]);
ylim(ylim+[max(ylim)-min(ylim)]/10*[0,1]+[-dy,0]);
axis equal
Xc=0;
S=sum(Mr(Mr>0.5))+sum(Mb(Mb>0.5));% 求出粒子总数
R=max(max(Xb-Xc)); % 求出炉膛半径坐标长度
Lb=sqrt(R^2-Xb.^2);  % 求出红色粒子受力权重系数矩阵
Lr=sqrt(R^2-Xr.^2);   % 求出蓝色粒子受力权重系数矩阵
Ti=title(['Times = ',num2str(0)],'Fontsize',18,...
   'Fontname','Times new roman');% 在图题处实时显示运行时间
YL=ylim;% 获取当前图形的 Y 轴范围
XL=xlim;% 获取当前图形的 X 轴范围
Po=get(gca,'Position');% 获取当前图形的位置
subplot(122);% 开第二子图
NL=size(Xr,2)+size(Xb,2);% 求出红色粒子的坐标矩阵和蓝色粒子坐标矩阵的总列数
xt(1:2:NL)=Xb(1,:);% 合成红色粒子和蓝色粒子的坐标矩阵第一行为一个行向量
xt(2:2:NL)=Xr(1,:);% 合成红色粒子和蓝色粒子的坐标矩阵第一行为一个行向量
yt(1:2:NL)=max(Yb.*Mb);% 求出最上层粒子的坐标
yt(2:2:NL)=max(Yr.*Mr);% 求出最上层粒子的坐标
pt=plot(xt,yt,'k');% 求出最上层粒子对应的曲线
ylim(YL);% 设置第二子图的 Y 轴范围与第一子图相同
xlim(XL);% 设置第二子图的 X 轴范围与第一子图相同
```

```
T=0;% 初始化时间
Mr(1,Mr(1,:)>0.5)=0;%设置最小层红色粒子为空(用 0 表示空粒子)
while S>0.5;%当粒子数大于 0.5 时程序保持运行状态，否则程序停止
    S=sum(Mr(Mr>0.5))+sum(Mb(Mb>0.5));%求出当前炉膛内的粒子总数
    [Mr,Mb]=pmovingg3(Mr,Mb,Lr,Lb);%计算炉膛内粒子的运动
    M1t=Mb;%置换当前红色粒子矩阵为 M1t
    M2t=Mr;%置换当前蓝色粒子矩阵为 M2t
    M1t(M1t<0.5)=nan;%用非 nan 来表示 M1t 中状态为 0 和边界-1
    M2t(M2t<0.5)=nan;%用非 nan 来表示 M2t 中状态为 0 和边界-1
    X1t=Xb.*M1t;%获得当前红色粒子的 X 坐标
    X2t=Xr.*M2t;%获得当前蓝色粒子的 X 坐标
    Y1t=Yb.*M1t;%获得当前红色粒子的 Y 坐标
    Y2t=Yr.*M2t;%获得当前蓝色粒子的 Y 坐标
    set(P1,'Xdata',X1t(:),'Ydata',Y1t(:));%更新红色粒子的坐标
    set(P2,'Xdata',X2t(:),'Ydata',Y2t(:));%更新蓝色粒子的坐标
    T=T+1;%累加计数器
    set(Ti,'string',['Times = ',num2str(T),', SUM = ',...
          num2str(S)]);%更新计数器和总粒子数目
    yt(1:2:NL)=max(Y1t);%计算最顶层的蓝色粒子 Y 坐标
    yt(2:2:NL)=max(Y2t);%计算最顶层的红色粒子 Y 坐标
    strq=['Ay_',num2str(T),'.txt'];
    set(pt,'Ydata',yt);%更新顶层粒子对应的曲线
    Mr(1,Mr(1,:)>0.5)=0;%设置最小层红色粒子为空(用 0 表示空粒子)
    pause(0.2); %程序停顿一下以体现出动画效果
end
```



其中用到了粒子移动实现的程序，即 pmovingg3.m，其保存于光盘中的\Ch22 文件夹下。

该程序中边界采用函数 plot 绘制的折线来表示，而炉内的粒子利用函数 plot 画实心的圆圈表示。首先根据炉腔形状设置初始粒子分布。然后利用粒子下落的规则计算不同时刻炉内粒子的分布情况，并实时地更新绘制粒子的 plot 函数中的 XData 和 YData 更新粒子的位置。上述程序执行后可以演示粒子下落的过程，顶层粒子逐渐下落。当时刻 t 等于 200 时，可以得到如图 22.25 所示的图形。

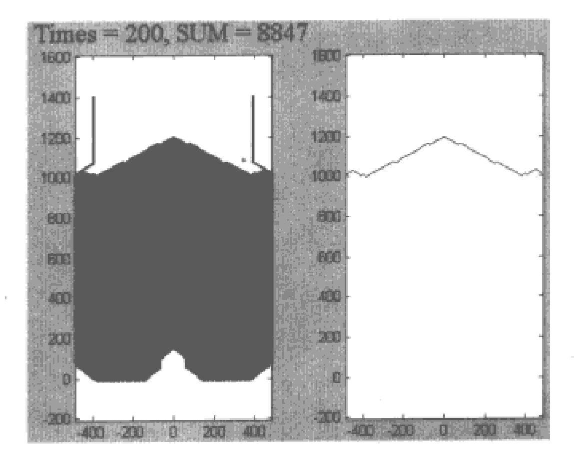


图 22.25　粒子下落过程的截图

22.7 小结

本章主要介绍了元胞自动机典型规则的程序实现。利用 MATLAB 矩阵化计算的优势，一些元胞自动机的规则可以并行实现。同时 MATLAB 的数据可视化可以方便地把元胞自动机的状态矩阵变化实时地显示出来。本章首先介绍了奇偶规则程序实现，利用 MATLAB 可以方便地实现这个规则。接下来给出了一个砂漏的程序模拟和细菌生长过程的模拟。在 Margolus 近邻上给出气体扩散过程的模拟。在蚂蚁规则的基础上研究了 3 种不同初始条件下的演化过程模拟。最后给出了六边形格子上的粒子运动模拟。

◆ ◆ ◆

第 23 章　晶体生长模拟

本章包括

- ◆ **随机布朗运动**　介绍随机布朗运动和随机行走模型。
- ◆ **扩散限制凝聚（DLA）**　给出扩散限制凝聚的定义及相应的程序实现。
- ◆ **随机吸附**　介绍该过程的编程思路，并给出树枝晶体生长的模拟。
- ◆ **随机向心吸附**　利用 MATLAB 模拟具有向心运动趋势的 DLA 模型。

前一章介绍了元胞自动机模型的一些基本规则，很多粒子的运动模型可以利用元胞自动机来模拟。而晶体的生长也可以描述为一些数学模型，从而可以利用这些模型来模拟晶体生长过程。在一定程度上，晶体生长过程可以看做是一种元胞自动机。本章介绍树枝状晶体生长过程的模拟方法，这其中要用到随机行走（Random Walk）模型、扩散限制凝聚（DLA）等。通过程序模拟这样的演化过程，可以从另外一个角度研究生长过程，一定程度上可以缩短研究周期，提高研究效率。

23.1　随机布朗运动

自从布朗发现花粉在液体中的随机运动现象后，人们开始利用这个现象来研究一些自然界中的过程模拟。1905 年爱因斯坦发表有关“布朗运动”的理论研究论文，迄今已有一个世纪。在这 100 年的时间里，布朗运动以及相关的随机行走（Random Walk）问题的研究有了长足发展，不仅对物理各领域有深远的影响，也在其他科学诸如化学、地理、生物以及经济学中广泛应用。而随机行走模型用来描述粒子以几率的形式在不同方向上游走。因为实际过程的复杂性，很多自然场合的粒子行为呈现出随机性。通过随机模型可以一定程度简化多体问题的研究。尽管这个过程是随机的，但是结果往往表现出一定的规律性。

在一个一维或者二维系统中可以使用一个随机数来控制粒子的运动方向，下面来举例说明这个过程的模拟。

与前一章介绍的蚂蚁规则相似，可以使用一个复数 z 来表示粒子的位置。这里考虑使用 x 和 y 来表示粒子当前位置的横纵坐标值。而运动方向的随机改变通过下面的语句来表达。

```
p1=0.2; % 向左运动的几率
p2=0.3; % 向上运动的几率
p3=0.1; % 向右运动的几率
p4=0.4; % 向下运动的几率
r=rand; % 产生一个随机数
if r<p1;
    dx=-1; % 水平方向的随机移动量
    dy=0; % 竖直方向的随机移动量
elseif r<p1+p2;
    dx=0; % 水平方向的随机移动量
    dy=-1; % 竖直方向的随机移动量
elseif r<p1+p2+p3;
```

```
    dx=1; % 水平方向的随机移动量
    dy=0; % 竖直方向的随机移动量
else
    dx=0; % 水平方向的随机移动量
    dy=1; % 竖直方向的随机移动量
end
```

其中 p1，p2，p3 和 p4 分别表示向左、向上、向右和向下运动的几率，如图 23.1 所示。这里要求 4 个几率值之和等于 1。

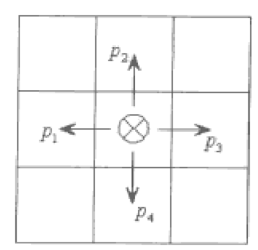


图 23.1 粒子随机运动的方向选择

为方便起见，把上面这段程序写为函数文件的形式，即 choose_D.m 文件，其调用格式为：

```
[dx,dy]=choose_D(p1,p2,p3,p4);
```

参数说明： dx 和 dy 是函数的输出参数，分别表示粒子在水平和竖直方向上的位置改变量。p1，p2，p3 和 p4 的含义如图 23.1 所示，使用时它们的输入顺序不能改变，否则会引起错误。

choose_D 函数可以用在随机行走或者后面将要介绍的扩散限制凝聚（DLA），可以直接根据几率数值来得到 dx 和 dy，利用它们可以计算粒子下一时间步的位置。下面给出利用函数 choose_D 实现随机行走的例子。

```
z=0; % 设置初始数值，这里利用一个复数来表示位置
rand('state',1);     % 初始化随机数的状态
ph=plot(z);hold on; % 绘图
T=1; % 计算时间的参数
z1=1; % 位置的中间变量
ti=title('time = 0','Fontsize',14,'Fontname','Times new roman'); % 在图题处实时显示时间
while abs(z1)>0.1; % 循环多次进行计算直至回到起点位置(0,0)
    [dx,dy]=choose_D(0.25,0.25,0.25,0.25); % 随机生成一个位移量
    z1=z+[dx+i*dy]; % 更新位置数值
    plot(real([z,z1]),imag([z,z1]));      % 连接 t 时刻和 t+1 时刻两点绘制线段
    z=z1; % 更新 z 的数值
    set(ti,'String',['time = ',num2str(T)]); % 更新图题处的时间数值
    T=T+1;    % 时间值更新
    pause(0.1); % 暂停一下，显示动画效果
end
```

执行上述程序后可以得到如图 23.2 所示的图形。选用不同随机函数的状态数可以得到不同的图案，图 23.3 给出了当状态数等于 5，即"rand('state',5);"时的图案。可见随着随机数选择的不同，可以得到不同的图案。可以证明，进行足够多次的计算后，粒子可以回到起始位置。

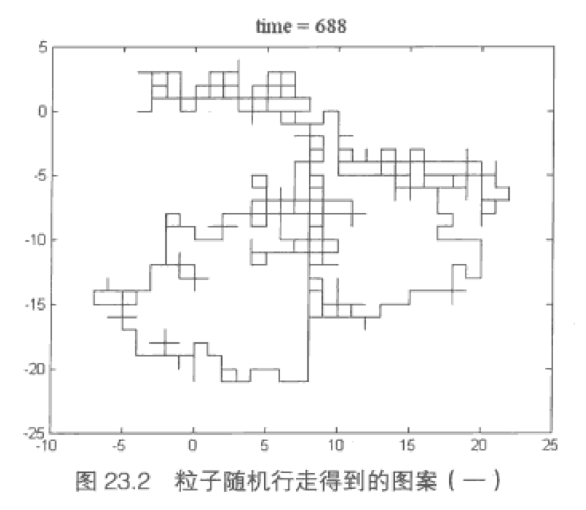


图 23.2　粒子随机行走得到的图案（一）

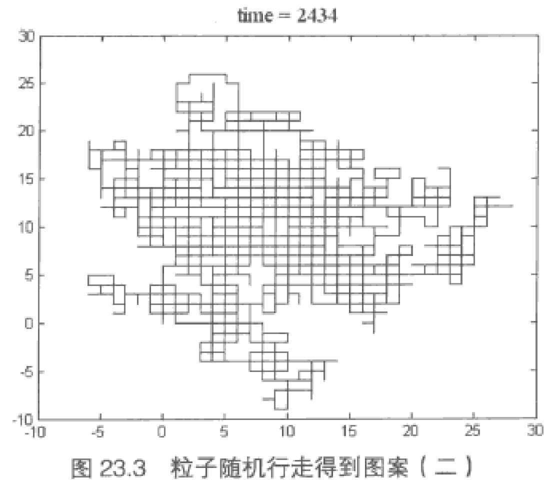


图 23.3　粒子随机行走得到图案（二）

前面介绍的是粒子沿着水平和竖直两个方向运动，下面给出粒子的运动方向可能是平面上的任意角度。相应的程序要比前面的程序简单一些。

```
z=0; % 设置初始数值，这里利用一个复数来表示位置
rand('state',5); % 初始化随机数的状态
ph=plot(z);hold on; % 绘图
T=1; % 计算时间的参数
z1=1; % 位置的中间变量
ti=title('time = 0','Fontsize',14,'Fontname','Times new roman'); % 在图题处实时显
示时间
while abs(z1)>0.1; % 循环多次进行计算直至回到起点位置(0,0)
    A=rand*pi*2;  % 生成一个随机角度
    z1=z+exp(i*A); % 更新位置数值
    plot(real([z,z1]),imag([z,z1])); % 连接 t 时刻和 t+1 时刻两点绘制线段
    z=z1; % 更新 z 的数值
    set(ti,'String',['time = ',num2str(T)]); % 更新图题处的时间数值
    T=T+1; % 时间值更新
    pause(0.1); % 暂停一下，显示动画效果
end
```

执行上面的程序后可以得到如图 23.4 所示的图形，这个图形可以认为是在二维平面上的布朗运动结果的模拟，这个图也是“一笔画”的结果，即轨迹不中断。可见这是一个运动方向完全随机的结果，粒子的轨迹也显得杂乱无章。与前面介绍的蚂蚁规则不同的是，布朗运动和前面的运动轨迹不相关，粒子运动方向不受之前状态矩阵的影响。

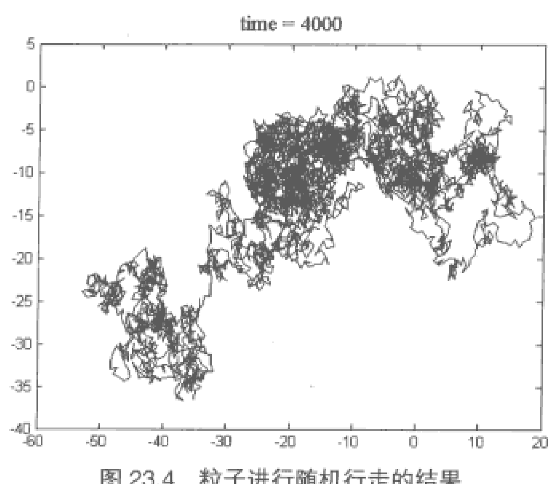


图 23.4 粒子进行随机行走的结果

分形布朗运动（Fractal Brown Motion，缩写为 FBM）曲线的物理定义是一维布朗运动的位移-时间坐标图，它们的物理意义和作图无关。分形布朗曲线的作图方法是选择点 A(x1,y1)为起点、点 B(x2,y2)为终点，选择分形布朗曲线的标度因子 H(0<H<1)，在线段 AB 的中点 C(x3,y3)取高斯分布（均值为 0，方差 DY=(X2-X1)/(1-2 ^ (2*H-2)) ^ 0.5）随机数 Y 作为偏移值，得 C''(x3,y3+Y)为分形布朗曲线上的点。然后再对线段 AC''和线段 C''B 重复上述过程，但是这时的高斯随机数方差为 DY/(2*2 ^ H)，即原方差除以 2 的 H+1 次方，这里的 H 称为标度因子，如此无限继续下去，就可以得到分形布朗曲线了，此方法属于随机中点位移法。在实际计算中不必取无限多次计算，取足够多次计算就可以了。下面是一个实现分形布朗曲线的程序。

```
function zp=Brown_motion(z1,z2,n,zp);  % 分形布朗运动的程序
N=8;  % 递归次数
H=0.5; % 分形布朗曲线的标度因子 H(0<H<1)
C=1-sqrt(2^(2*H-2)); % 系数因子
if n<N;
   zc=mean([z1,z2]); % 计算中点坐标值
   DY=real(z2-z1)/C; % 方差 DY
   if n>0;
       DY=DY/(2*2^H); % 系数因子
   end
   Y=normrnd(0,sqrt(DY)); % 计算偏移量
   z3=zc+Y*i;
   zp=[zp,z3]; % 更新 zp 的数据
   zp=Brown_motion(z1,z3,n+1,zp); % 进行递归计算
   zp=Brown_motion(z3,z2,n+1,zp); % 进行递归计算
else
   [X,K]=sort(real(zp)); % 按横坐标从小到大排序得到排序序号 K
   zp=zp(K);             % 排列数据的顺序
end
```

这是一个函数文件，其调用格式为：

```
zp=Brown_motion(z1,z2,n,zp);
```

参数说明： 输出量 zp 是数据点的坐标。z1 和 z2 是开始时的两个端点坐标，其可以选择复数值。n 是控制递归过程的参数。输入量 zp 是用于在传递递归计算的过程中传递坐标数据。

利用下面的语句可以调用前面的 Brown_motion 函数计算分形布朗运动的曲线。

```
rand('state',0); % 设置随机数的状态
z=Brown_motion(0,1,0,[0,1]); % 计算分形布朗运动的曲线数据点
plot(z); % 绘图
```

上面语句输出图形如图 23.5 所示。

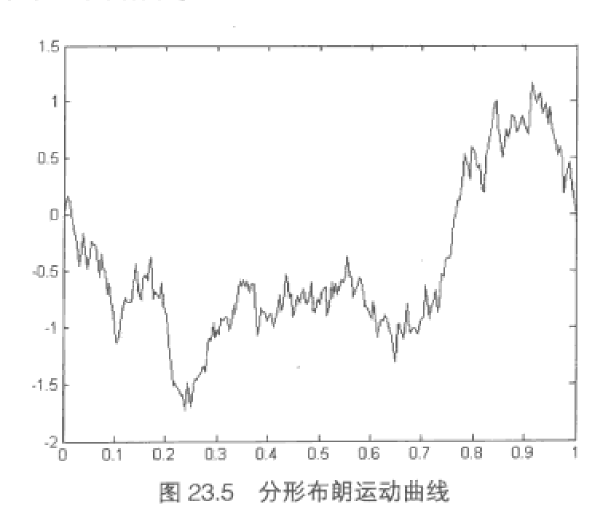


图 23.5 分形布朗运动曲线

23.2 扩散限制凝聚（DLA）

早在 1981 年，Witten 和 Sander 两位学者在研究悬浮于大气中的烟灰、金属粉末或者烟尘的扩散凝聚问题时，提出了扩散受限制的凝聚模型（Diffusion Limited Aggregation，缩写为 DLA），目前这个模型被应用到不同领域。

DLA 模型可以模拟下面的电解过程：把负电极放置在带有正的金属离子的电解液中，金属离子在电解液中将会进行扩散，当它们遇到负电极时会被负电极所吸引。此外这些例子也可能在已经沉淀的凝聚体上发生粘贴而被固定。如此下去，一系列复杂坚硬的树枝状结构将会被堆积出来。对于这个沉积、吸附过程可以使用数值方法进行求解。

DLA 模拟的数学模型可以描述为：在一个阵列形点阵上面，在中心位置处布置一个固定的粒子作为初始状态，在距离该点一定范围之外产生粒子，这个粒子在未被吸附之前将进行随机行走运动，直至它被固定的粒子所吸附才停止下来。这里吸附的原则是：在 4 个最近邻（指上、下、左、右格点）之中存在一个固定的粒子，则运动的粒子将被吸附，其停止运动。接下来产生下一个粒子，并按上面的规则进行计算。如此下去将会得到一个凝聚集图案。

需要指出的是，如果粒子在随机行走时超出状态矩阵所限制的范围，需要考虑对应的处理方案。这里可能用到的方案有两个：一是采用周期边界条件，这样粒子就不会逃逸，直至吸附到固定的粒

子上才能产生下一个粒子；二是采用吸收边界条件，这样粒子超出边界之后将放弃这个粒子而去生成下一个粒子。

根据上面的分析，建立相应的实现程序。首先给出采用周期边界条件来建立 DLA 模型，这里使用白色（矩阵元素值等于 1）表示该格子位置上没有粒子，而使用黑色（矩阵元素等于 0）表示格子位置上有粒子。这样，运动粒子的 4 个最近邻之和小于 3.5 就表示周围至少有一个固定的粒子，从而粒子被吸附住。

相应的 MATLAB 程序如下：

```
rand('state',0); % 设置随机数的状态值
set(gcf,'DoubleBuffer','on'); % 设置渲染效果
N=256;     % 生成状态矩阵大小的控制参数
S=ones(N); % 生成状态矩阵 S
S(N/2,N/4:N/4*3)=0; % 设置状态矩阵的初始值
Ii=imshow(S);      % 显示状态矩阵
ti=title('time = 0','Fontsize',14,'Fontname','Times New Roman'); % 显示时间
T=0; % 记录时间的参数
for k=1:4350; % 循环计算
    xt=4;    % 产生粒子的位置
    yt=N/2;  % 产生粒子的位置
    Ss=0;      % 控制下面循环是否终止的参数
    while Ss<1; % 计算当前粒子的吸附过程
        [dx,dy]=choose_D(0.25,0.25,0.25,0.25); % 计算粒子位移量
        xt=xt+dx; % 计算下一时刻粒子的位置
        yt=yt+dy; % 计算下一时刻粒子的位置
        yt=mod(yt-1,N)+1; % 利用周期边界条件限制粒子坐标位置
        xt=mod(xt-1,N)+1; % 利用周期边界条件限制粒子坐标位置
        xn=xt+[-1,1,0,0]; % 计算最近邻的坐标
        yn=yt+[0,0,-1,1]; % 计算最近邻的坐标
        yn=mod(yn-1,N)+1; % 对近邻进行周期边界条件处理
        xn=mod(xn-1,N)+1; % 对近邻进行周期边界条件处理
        Ind=sub2ind([N,N],xn,yn); % 转化脚标为索引
        if sum(S(Ind))<3.5;        % 判断近邻是否有粒子
            S(xt,yt)=0; % 固定当前运动的粒子
            Ss=2;      % 把 Ss 赋值为 2，从而可以退出循环
        end
    end
    set(Ii,'CData',S); % 显示当前状态矩阵
    T=T+1;          % 累加时间参数
    set(ti,'String',['time = ',num2str(T)]); % 更新时间参数
    pause(0.01); % 暂停一下，显示动画效果
end
```

上述程序输出图形如图 23.6 所示。这里初始时的状态矩阵的中间有一条水平线上的粒子是静止的，在固定位置的粒子随机运动，在最初的直线上进行随机吸附。最后在各个方向上都长出了树枝状分支，在上、下两个方向树枝生长的长度较长。随着固定粒子数目的增加，吸附的速度也要增加。

如果把语句“S(N/2,N/4:N/4*3)=0;”改为“S(N/2:N/4*3,N/2)=0;”，可以研究状态矩阵的初始固定粒子成竖直方向线段时 DLA 演化的模拟，所得图形如图 23.7 所示。

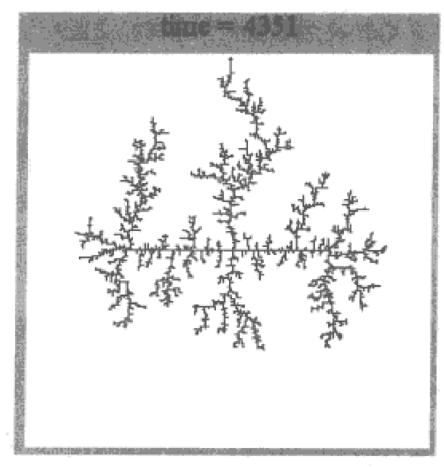


图 23.6　周期边界条件下 DLA 模型生成的图案

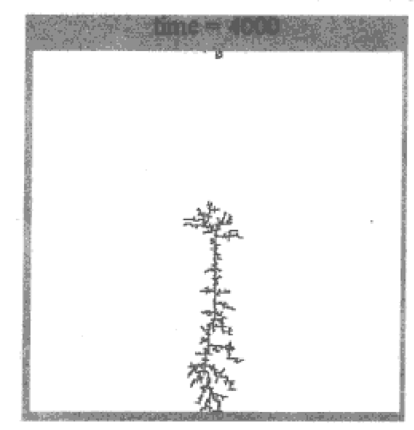


图 23.7　周期边界条件下 DLA 模型得到的图案

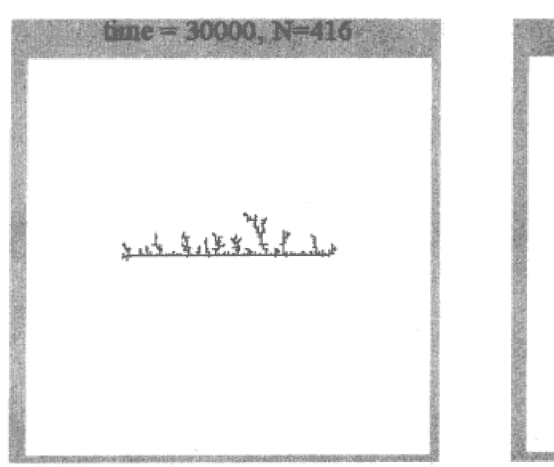
下面考虑吸收边界条件的实现：在状态矩阵 S 中，以 $x=2$，$x=N-1$，$y=2$ 和 $y=N-1$ 四条直线作为边界，粒子逸出边界（包含边界）将产生下一个粒子，而其他部分可以和周期边界条件采取相同的实现方法。相应的 MATLAB 程序如下：

```
rand('state',0); % 设置随机数的状态
set(gcf,'DoubleBuffer','on'); % 设置图形窗口的渲染效果
N=256;      % 生成状态矩阵大小的控制参数
S=ones(N);  % 生成状态矩阵 S
S(N/2,N/4:N*3/4)=0; % 设置状态矩阵的初始值
Ii=imshow(S);       % 显示状态矩阵
ti=title(['time = 0',', N=0'],'Fontsize',14,'Fontname','Times New Roman'); % 显示时间
T=0; % 记录时间的参数
for k=1:30000; % 循环计算
    xt=4;    % 产生粒子的位置
    yt=N/2; % 产生粒子的位置
    Ss=0;   % 控制下面循环是否终止的参数
    while Ss<1;  % 计算当前粒子的吸附过程
        [dx,dy]=choose_D(0.25,0.25,0.25,0.25); % 计算粒子位移量
        xt=xt+dx; % 计算下一时刻粒子的位置
        yt=yt+dy; % 计算下一时刻粒子的位置
        if xt<2.5|xt>N-1.5||yt<2.5|yt>N-1.5; % 判断粒子是否逸出边界
```

```
            Ss=2; % 把 Ss 赋值为 2，从而可以退出循环
        end
        if [S(xt+1,yt)+S(xt-1,yt)+S(xt,yt+1)+S(xt,yt-1)]<3.5; % 判断近邻是否有粒子
            S(xt,yt)=0; % 固定当前运动的粒子
            Ss=2;     % 把 Ss 赋值为 2,从而可以退出循环
        end
    end
    set(Ii,'CData',S); % 显示当前状态矩阵
    T=T+1;          % 累加时间参数
    set(ti,'String',['time = ',num2str(T),', N=',num2str(sum(sum(1-S))-[N/2+1])]);
% 更新时间参数
    pause(0.01); % 暂停一下，显示动画效果
end
```

上述程序执行后可以得到如图 23.8 左图所示的图形。可见在初始固定粒子下面很少出现粒子吸附的现象，绝大部分粒子在初始线段上面就被“挡住”了。前面周期边界条件理解为粒子可以从上面边界运动至初始线的下侧，这样上下生长规模相似了。另外在图 23.8 左图的图题处 time 数值对应着在粒子源点处产生的粒子总数，N 表示被吸附的粒子总数，而 N 和 time 二者的数值相差很多，这表明大部分粒子都逸出边界之外了。

当把语句“S(N/2,N/4:N*3/4)=0;”换为“S(N/2:N*3/4,N/2)=0;”时，可以得到如图 23.8 右图所示的图形。可见对于吸收边界条件，树枝状结果的生长方向是朝向粒子源所在方向的。



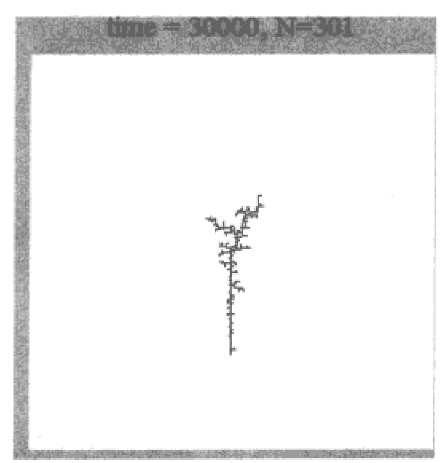


图 23.8　吸收边界条件下的 DLA 模型的演化图案

在前面给出各向同性（粒子向 4 个方向运动的几率相同）时 DLA 模型的实现，下面考虑粒子向下运动几率较大时的情况，即 $p_1=0.2$，$p_2=0.2$，$p_3=0.2$ 和 $p_4=0.4$ 时的 DLA 模型程序。与前面不同的是，产生粒子的源是在从上侧数第 4 行的水平直线上，具体位置是随机的。实现程序如下：

```
rand('state',0); % 设置随机数的状态
set(gcf,'DoubleBuffer','on');% 设置图形窗口的渲染效果
N=400; % 生成状态矩阵大小的控制参数
M=200; % 生成状态矩阵大小的控制参数
S=ones(M,N); % 生成状态矩阵 S
S(end-2,:)=0; % 设置状态矩阵的初始值
Ii=imshow(S); % 显示状态矩阵
```

```
ti=title(['time = 0',', N=0'],'Fontsize',14,'Fontname','Times New Roman'); % 显示时间
T=0; % 记录时间的参数
for k=1:30000; % 循环计算
    xt=4; % 产生粒子的位置
    yt=2+round((N-4)*rand); % 产生粒子的位置
    while 1; % 计算当前粒子的吸附过程
        [dx,dy]=choose_D(0.2,0.2,0.2,0.4); % 计算粒子位移量
        xt=xt+dy; % 计算下一时刻粒子的位置
        yt=yt+dx; % 计算下一时刻粒子的位置
        if xt<3.5|xt>M-1.5||yt<3.5|yt>N-3.5; % 判断粒子是否逸出边界
            break; % 跳出当前循环
        end
        if [S(xt+1,yt)+S(xt-1,yt)+S(xt,yt+1)+S(xt,yt-1)]<3.5; % 判断近邻是否有粒子
            S(xt,yt)=0; % 固定当前运动的粒子
            break; % 跳出当前循环
        end
    end
    set(Ii,'CData',S); % 显示当前状态矩阵
    T=T+1; % 累加时间参数
    set(ti,'String',['time = ',num2str(T),', N=',num2str(sum(sum(1-S))-N)]); % 更新时间参数
    pause(0.01); % 暂停一下，显示动画效果
end
```

执行上述程序输出图形如图 23.9 所示。可见产生的粒子很快被吸附在固定的粒子上。

此外，可以考虑周期边界条件的模拟。这里给出 $p_1=0.3$，$p_2=0$，$p_3=0.3$ 和 $p_4=0.4$ 时 DLA 模型得到的结果，如图 23.10 所示。相应程序保存在光盘的\Ch23 文件夹下的 DLA_period2.m 文件中。在这种情况下粒子只能沿着向下、向左和向右 3 个方向运动。

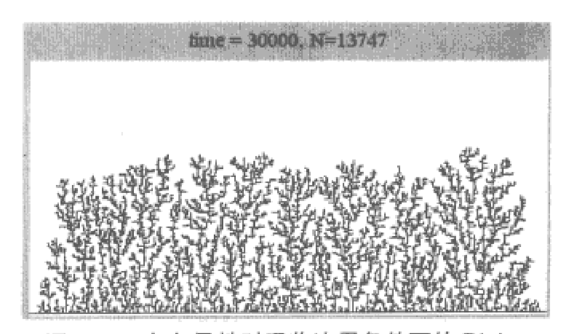


图 23.9 各向异性时吸收边界条件下的 DLA 模型演化的图案

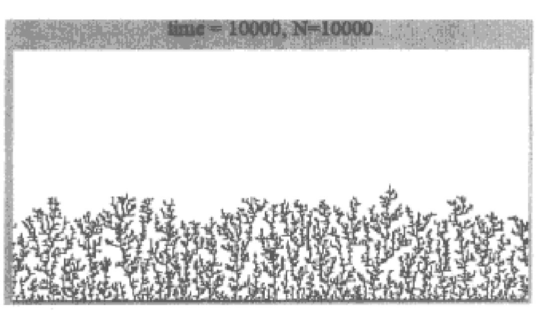


图 23.10 各向异性周期边界条件下的 DLA 模型演化的图案

下面研究粒子源呈圆形的 DLA 模型，这里采用周期边界条件，初始时中心的圆形区域内有一些固定的粒子。相应的 MATLAB 程序如下：

```
rand('state',0); % 设置随机数的状态
set(gcf,'DoubleBuffer','on');% 设置图形窗口的渲染效果
N=256; % 生成状态矩阵大小的控制参数
M=256; % 生成状态矩阵大小的控制参数
S=ones(M,N); % 生成状态矩阵 S
[x,y]=meshgrid(linspace(-N/2,N/2,N),linspace(-M/2,M/2,M));
S(abs(x+i*y)<N/16)=0; % 设置状态矩阵的初始值
```

```
S0=sum(sum(1-S));    % 计算初始时固定粒子总数
Ii=imshow(S); % 显示状态矩阵
ti=title(['time = 0',',  N=0'],'Fontsize',14,'Fontname','Times New Roman'); % 显
示时间
T=0; % 记录时间的参数
for k=1:8000; % 循环计算
    z=round([N-20]*exp(i*rand*pi*2));
    xt=real(z); % 产生粒子的位置
    yt=imag(z); % 产生粒子的位置
    while 1; % 计算当前粒子的吸附过程
        [dx,dy]=choose_D(0.25,0.25,0.25,0.25); % 计算粒子位移量
        xt=xt+dy; % 计算下一时刻粒子的位置
        yt=yt+dx; % 计算下一时刻粒子的位置
        xn=mod(xt-1+[-1,1,0,0],M)+1; % 进行周期边界条件处理
        yn=mod(yt-1+[0,0,-1,1],N)+1; % 进行周期边界条件处理
        Ind=sub2ind([M,N],xn,yn);
        if sum(S(Ind))<3.5; % 判断近邻是否有粒子
            S(mod(xt-1,M)+1,mod(yt-1,N)+1)=0; % 固定当前运动的粒子
            break; % 跳出当前循环
        end
    end
    set(Ii,'CData',S); % 显示当前状态矩阵
    T=T+1 % 累加时间参数
    set(ti,'String',['time = ',num2str(T),', N=',num2str(sum(sum(1-S))-S0)]); % 更
新时间参数
    pause(0.01); % 暂停一下，显示动画效果
end
```

首先在状态矩阵的中央布置一些固定的粒子，它们的分布形状呈现一个圆形，然后在一个圆形的位置上随机产生粒子并进行随机布朗运动，直至其与固定的粒子发生粘贴才产生下一个粒子。多次计算之后可以得到 DLA 演化图案。执行上述程序输出的图形如图 23.11 所示。此外还可以考虑吸收边界条件下 DLA 模型的实现。

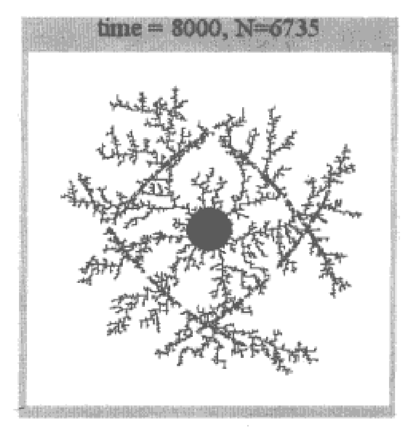


图 23.11　周期边界条件和圆形粒子源下 DLA 模型得到的图案

这里给出粒子生长受限的一个例子，即当粒子吸附到的树枝长到一定高度时，相应的分支将被切掉。切去分支的程序被写为一个函数文件形式，保存在光盘\Ch23 文件夹中，文件名为 cuth.m。因为凝聚体的分支是不规则的形状，所以函数文件 cuth.m 需要使用递归算法编写。在这个程序中，计算使用数值 1 表示该位置有粒子，而 0 表示该位置没有粒子。在显示粒子分布的时候，用 1-S

来显示（S 是状态矩阵），这样白色表示该格点处空白，黑色表示粒子被固定。而主程序的内容如下：

```
S=zeros(400,500); % 生成状态矩阵
S(end,:)=1;       % 设置状态矩阵中最下面一行元素等于 1
A=1;B=1;X=0.8;
rand('state',0); % 设置随机数的状态数
subplot(121);Ii=imshow(1-S,[]); % 显示状态矩阵
T1=title(['times = 1',', total particle=',num2str(sum(S(:)))],...
    'Fontname','times new roman','fontsize',14); % 显示时间与粒子总数
r=rand(1,500);
subplot(122);P1=plot(sum(S,2)/size(S,2),1:size(S,1),'r');% 绘制各行的密度值曲线
set(gca,'Position',[0.57,0.35,0.33,0.36],'YDir','reverse'); % 设置坐标轴属性
xlim([0,max(sum(S,2)/size(S,2))]); % 设置 X 轴的范围
ylabel('\ith','fontname','times new roman','fontsize',14);   % Y 轴标注
xlabel('\it\rho','fontname','times new roman','fontsize',14); % X 轴标注
title('{\it\rho} ({\ith})','fontname','times new roman','fontsize',14); % 加注图
题
set(gcf,'DoubleBuffer','on'); % 设置图形窗口的渲染效果
[L1,L2]=size(S); % 返回状态矩阵的行数 L1 和列数 L2
N=500;H=1;   % 初始化参数：粒子总数 N 和时间参数 H
h=150;    % 设置截顶高度
scale=0.5; % 设置剪切系数
while N<20000;
    R1=2+round([L1-3]*rand); % 随机产生粒子的坐标
    R2=2+round([L2-4]*rand); % 随机产生粒子的坐标
    flag=0; % 控制循环停止的参数
    while R1<L1&R1>1&R2<L2&R2>1&flag==0; % 验证粒子在状态矩阵内部且粒子未被吸附
        he=S(R1,R2-1)+S(R1,R2+1)+S(R1+1,R2); % 计算左、下和右方位的近邻
        if he>0.5; % 判断近邻中有无固定粒子
            S(R1,R2)=1; % 运动粒子被吸附
            flag=1;     % 标记粒子已经被吸附
        else
            ra=rand; % 粒子进行随机移动的分量
            rb=rand; % 粒子进行随机移动的分量
            R1=R1+(ra>=0.5)-(ra<0.5); % 计算下一时刻粒子的位置坐标
            R2=R2+(rb>=0.5)-(rb<0.5); % 计算下一时刻粒子的位置坐标
        end
    end
    sS=sum(S,2); % 对行所有元素求和
    Se=find(sS);Se=min(Se); % 找出有粒子的最高一行
    if Se==[size(S,1)-h]; % 判断高度是否达到截顶高度
        Sx=find(S(Se,:)); % 找出最高点粒子的横坐标
        S=cuth(S,h,Se,Sx,scale); % 切去最高点粒子所在的分支
    end
    set(Ii,'CData',1-S); % 显示状态矩阵
    N=sum(S(:)); % 计算粒子总数
    H=H+1;     % 累计时间值
    set(P1,'XData',sum(S,2)/size(S,2)); % 更新密度曲线数据
   set(T1,'string',['times = ',num2str(H),', total
particle=',num2str(sum(S(:)))]);% 更新时间和粒子总数
   pause(0.02); % 暂停一下，显示动画效果
end
```

首先在状态矩阵的最下面一层布置固定的粒子。然后在初始固定粒子上方随机产生一个粒子并

进行随机布朗运动，直至其与固定的粒子发生粘贴才产生下一个粒子。当某一分支达到截定高度后，该分支将会被切去。多次计算之后就可以得到 DLA 演化图案。上述程序可以不停地执行，直至强行终止它。其中一个截图如图 23.12 所示。左图实时显示粒子分布状态的变化，右图是各行粒子密度对应的曲线。在右图中，由于存在截顶限制，所以非零密度值对应的曲线一定在[250,400]这个范围内，也就是说粒子的高度被限制在 150 之内。

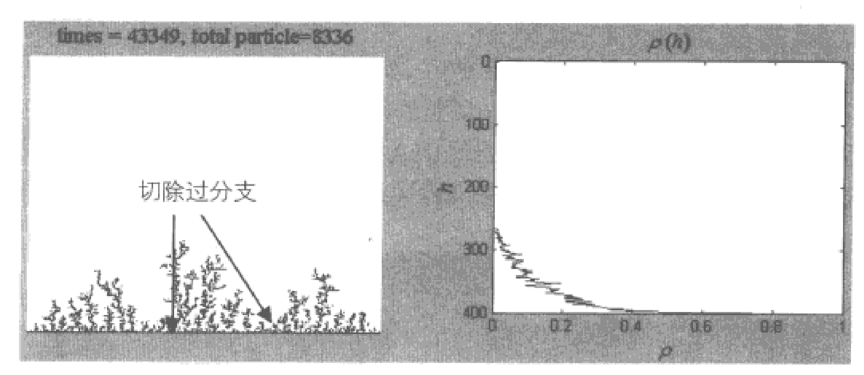


图 23.12 带有截顶限制的 DLA 模型

23.3 随机吸附

前一节介绍了不同类型的 DLA 模型，这里考虑一种带有吸附几率的 DLA 模型。也就是说粒子遇到固定粒子时以一定的几率吸附到固定的粒子上。这里吸附几率取 0.04，而粒子的运动方向取格子的对角线，边界条件选择周期边界条件。

相应的实现程序如下：

```
rand('state',0); % 设置随机数的状态
set(gcf,'DoubleBuffer','on');% 设置图形窗口的渲染效果
N=500; % 生成状态矩阵大小的控制参数
M=200; % 生成状态矩阵大小的控制参数
S=ones(M,N); % 生成状态矩阵 S
S(end,:)=0;   % 设置状态矩阵的初始值
Ii=imshow(S); % 显示状态矩阵
ti=title(['time = 0',', N=0'],'Fontsize',14,'Fontname','Times New Roman'); % 显
示时间
T=0; % 记录时间的参数
while sum(S(100,:))>N-0.5; % 循环计算
    xt=4; % 产生粒子的位置
    yt=2+round((N-4)*rand); % 产生粒子的位置
    while 1; % 计算当前粒子的吸附过程
        [dx,dy]=choose_D(0.3,0,0.3,0.4); % 计算粒子位移量
        xt=xt+dy; % 计算下一时刻粒子的位置
        yt=yt+dx; % 计算下一时刻粒子的位置
        xn=mod(xt-1+[-1,1,0,0],M)+1; % 进行周期边界条件处理
        yn=mod(yt-1+[0,0,-1,1],N)+1; % 进行周期边界条件处理
        Ind=sub2ind([M,N],xn,yn);
        if sum(S(Ind))<3.5; % 判断近邻是否有粒子
            S(mod(xt-1,M)+1,mod(yt-1,N)+1)= [rand>0.04]; % 固定当前运动的粒子
            break; % 跳出当前循环
```

```
        end
    end
    set(Ii,'CData',S); % 显示当前状态矩阵
    T=T+1; % 累加时间参数
    set(ti,'String',['time = ',num2str(T),', N=',num2str(sum(sum(1-S))-N)]); % 更
新时间参数
    pause(0.01); % 暂停一下，显示动画效果
end
```

首先在状态矩阵的最下面一层布置固定的粒子，然后在初始固定粒子上方的一条水平线上随机产生一个粒子并进行随机布朗运动，直至其与固定的粒子发生粘贴才产生下一个粒子。其中对各个时刻的图案实时显示。多次计算之后就可以得到 DLA 演化图案。执行上述程序输出图形如图 23.13 所示。对于吸收边界条件的程序，可以进行类似编写。此外，程序中的吸附系数可以换用其他介于$(0,1]$的数。

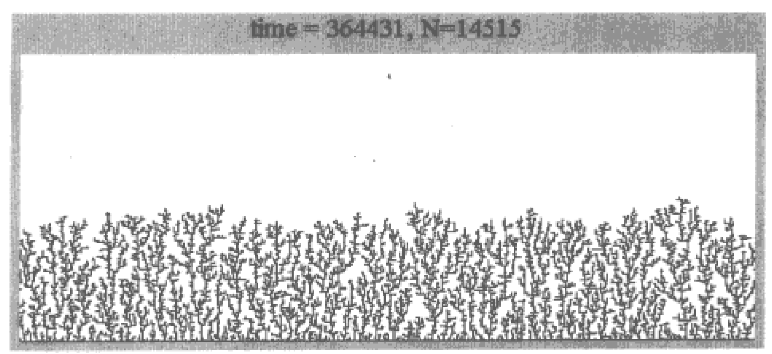


图 23.13　带有随机吸附的 DLA 模型演化的图案

23.4　随机向心吸附

下面考虑一种具有向心运动趋势的 DLA 模型。这里考虑粒子在一个圆形的粒子源上随机地出现。初始时的固定粒子分布为圆环形，且分布在粒子状态矩阵的中心。粒子的运动方向具有向心运动的趋势，其中向左、向上、向右和向下运动的几率分别表示为 p_1， p_2， p_3和 p_4，它们可以表示为下面的形式。

$$p_1=\begin{cases}k\cos^2\theta+(1-k)r_1, & \cos\theta<0\\ 0+(1-k)r_1, & \cos\theta\geq 0\end{cases} \tag{23-1}$$

$$p_2=\begin{cases}k\sin^2\theta+(1-k)r_2, & \sin\theta>0\\ 0+(1-k)r_2, & \sin\theta\leq 0\end{cases} \tag{23-2}$$

$$p_3=\begin{cases}k\cos^2\theta+(1-k)r_3, & \cos\theta>0\\ 0+(1-k)r_3, & \cos\theta\leq 0\end{cases} \tag{23-3}$$

$$p_4=\begin{cases}k\sin^2\theta+(1-k)r_4, & \sin\theta<0\\0+(1-k)r_4, & \sin\theta\geq 0\end{cases} \tag{23-4}$$

其中 k 是向心运动的几率比例。r_1，r_2，r_3 和 r_4 表示 4 个随机数，它们的总和是 1。在粒子每次运动之前，先利用上面的 4 个公式计算 4 个方向的几率。为了更好地了解上面 4 个等式的含义，可以参考图 23.14 所示。

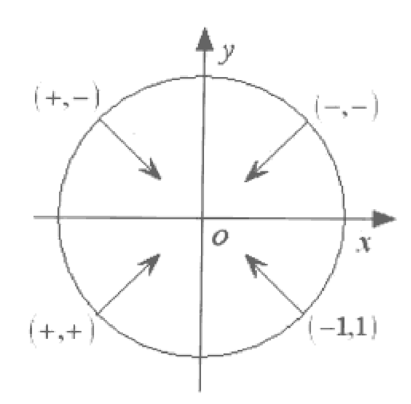


图 23.14　向心趋势吸附的 DLA 模型

这里使用周期边界条件编写程序，具体内容如下：

```
rand('state',0);
set(gcf,'DoubleBuffer','on');
N=256; % 生成状态矩阵大小的控制参数
S=ones(N);      % 生成状态矩阵 S
R=round([N-1]/2); % 圆形粒子源的半径
[x,y]=meshgrid(linspace(-N/2,N/2,N));
S(abs(x+i*y)<N/32)=0; % 设置状态矩阵的初始值
S(abs(x+i*y)<N/40)=1; % 设置状态矩阵的初始值
Ii=imshow(S); % 显示状态矩阵
ti=title('time = 0','Fontsize',14,'Fontname','Times New Roman'); % 显示时间
T=0;   % 记录时间的参数
sc=0.6; % 相当于系数公式中的 k
for k=1:6000;     % 循环计算
   rn=rand*2*pi; % 产生一个随机的角度值
   xt=round(R*cos(rn)); % 产生粒子的位置
   yt=round(R*sin(rn)); % 产生粒子的位置
   Ss=0;      % 控制下面循环是否终止的参数
   while Ss<1; % 计算当前粒子的吸附过程
       r4=rand(1,4); % 产生 4 个随机数
       r4=r4/sum(r4); % 归一化随机数
       p1=[cos(rn)<0]*cos(rn)^2*sc+(1-sc)*r4(1); % 根据角度分布情况确定各方向运动的几率
       p2=[sin(rn)>0]*sin(rn)^2*sc+(1-sc)*r4(2);  % 根据角度分布情况确定各方向运动的
几率
       p3=[cos(rn)>0]*cos(rn)^2*sc+(1-sc)*r4(3); % 根据角度分布情况确定各方向运动的几率
       p4=[sin(rn)<0]*sin(rn)^2*sc+(1-sc)*r4(4);  % 根据角度分布情况确定各方向运动的
几率
       [dx,dy]=choose_D(p1,p2,p3,p4); % 计算粒子位移量
       xt=xt+dy; % 计算下一时刻粒子的位置
       yt=yt+dx; % 计算下一时刻粒子的位置
```

```
        yt=mod(yt-1,N)+1; % 利用周期边界条件限制粒子坐标位置
        xt=mod(xt-1,N)+1; % 利用周期边界条件限制粒子坐标位置
        xn=xt+[-1,1,0,0]; % 计算最近邻的坐标
        yn=yt+[0,0,-1,1]; % 计算最近邻的坐标
        yn=mod(yn-1,N)+1; % 对近邻进行周期边界条件处理
        xn=mod(xn-1,N)+1; % 对近邻进行周期边界条件处理
        Ind=sub2ind([N,N],xn,yn); % 转化脚标为索引
        S(xt,yt)=[sum(S(Ind))>3.5]; % 判断近邻是否有粒子，如果有则固定当前运动的粒子
        Ss=2*[1-S(xt,yt)]; % 把 Ss 赋值为 2，从而可以退出循环
    end
    set(Ii,'CData',S); % 显示当前状态矩阵
    T=T+1; % 累加时间参数
    set(ti,'String',['time = ',num2str(T)]); % 更新时间参数
    pause(0.01); % 暂停一下，显示动画效果
end
```

首先在状态矩阵的中央区域布置一些固定的粒子，它们呈圆环形分布。然后在圆圈上随机产生一个粒子并进行随机布朗运动，各个粒子运动方向具有向心运动的趋势，直至其与固定的粒子发生粘贴才产生下一个粒子。其中对各个时刻的图案实时显示。多次计算之后就可以得到 DLA 演化图案。执行上述程序后可以得到如图 23.15 所示的图形，这个图形的分布是各方向随机运动和向心运动的总和。

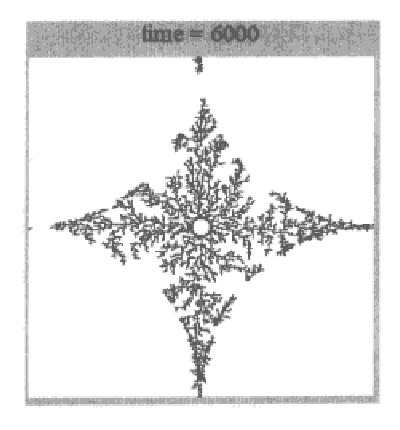


图 23.15　具有向心运动趋势的 DLA 模型模拟图案

23.5　小结

本章主要介绍了树枝状晶体生长 MATLAB 数值模拟的方法。其中主要理论基础是随机布朗运动的模型。随机布朗运动可以直接用来模拟随机行走模型。在随机布朗运动的基础上，实现了扩散限制凝聚模型。利用扩散限制凝聚模型可以模拟一些树枝状晶体生长的过程。本章还介绍了周期边界条件、吸收边界条件的实现方法，以及不同随机运动的形式，如随机吸附模型、具有向心运动趋势的 DLA 模型。利用这些基本的模块，可以模拟不同条件下树枝状晶体生长的过程。

◆　◆　◆

第 24 章　光学现象模拟

本章包括

- **网格上的鱼眼**　介绍鱼眼效果的模拟方法，给出不同形式鱼眼网格的绘制程序。
- **计算全息编码及再现程序**　介绍全息编码的程序实现方法。
- **光的等厚干涉**　结合空气劈尖介绍等厚干涉的模拟。
- **杨氏双缝干涉**　介绍杨氏双缝干涉的原理，同时给出相应的模拟程序。
- **牛顿环**　介绍一个模拟牛顿环的用户界面程序示例。

在很多光学问题研究中，使用了数值模拟来实现这个过程或者结果。随着目前学科的细化，数值模拟的重要性在不同学科研究中日益突出。具有快速、节省资源资金、富于变化等特点。光学在信息、测量、微观结构研究等前沿领域有着重要应用。本章介绍基本光学现象的模拟，其中包括鱼眼效果的模拟、全息、干涉现象以及牛顿环的模拟。

24.1　网格上的鱼眼

虚拟景观的研究目前已经成为计算机图形学的一个研究热点，其原因就是它具有极大的实际应用价值。特别是随着计算机、网络以及电子商务的快速发展，虚拟景物在环境介绍、旅游和一些商业活动中具有重要的作用。如果使用普通相机镜头进行照片的拍摄，即使利用广角镜也需要拍摄多幅照片，然后通过一些复杂的图像处理方法才能得到建立在曲面（如柱面或者球形表面等）表面的全景图，这是一个非常烦琐的工作。而使用鱼眼镜头就可以拍摄 180° 甚至是大于 180° 的景物。鱼眼镜头的基本原理如图 24.1 所示。

鱼眼镜头成像原理图如图 24.2 所示，如果把照相机放置到坐标系的原点上，而拍摄方向选择沿 Z 轴正方向（光轴所在方向），这样成像面就是一个和 XOY 平行的平面。如此，图 24.2 就简化成了二维 XOZ 坐标的情况。鱼眼成像的基本原理可以描述如下：假设 S 是以原点 O 为焦点、MN 为准线的抛物线，该抛物线与 X 轴相交于 A 和 B 两点，令 $r=|OA|$，则抛物线方程可以表示为：

$$(x-x_c)^2+(z-z_c)^2=\left[r-(z-z_c)\right]^2 \tag{24-1}$$

其中 O 点的坐标为 (x_c,z_c)，若 O 点正好落在坐标原点上，那么 $x_c=z_c=0$。这里取 $z_c=0$，那么等式（24-1）可以改写为：

$$(x-x_c)=-2rz+r^2 \tag{24-2}$$

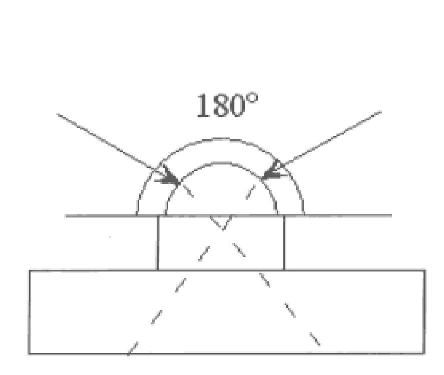


图 24.1 鱼眼镜头的基本原理

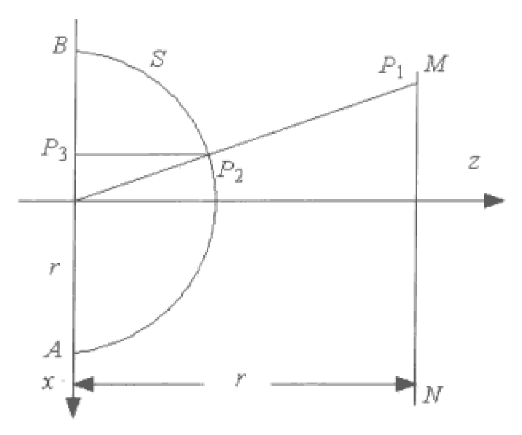


图 24.2 鱼眼镜头成像原理图

等式（24-2）是一个抛物线方程。对于景物上一点 P_1，连接 O 点和 P_1 点相交于抛物线上的 P_2 点，通过 P_2 点作垂直于 X 轴的线段，与 X 轴相交于 P_3 点，P_3 便是 P_1 点通过鱼眼镜头拍摄成的像点。若鱼眼镜头能够拍摄 180° 的场景，则景物点的像点将会布满 X 轴上的线段 AB。

对于真实的三维情况，抛物线将变为旋转抛物面，即：

$$(x-x_c)^2+(y-y_c)^2+(z-z_c)^2=\left[r-(z-z_c)\right]^2 \tag{24-3}$$

类似于前面等式（24-1）和等式（24-2）之间的推导，可以得出下面的关系式：

$$(x-x_c)^2+(y-y_c)^2=-2rz+r^2 \tag{24-4}$$

三维成像过程和二维成像过程相似。利用关系式（24-4）就可以得到一些图形的鱼眼效果图。

```
N=30;                                      % 网格采样点数
figure;set(gcf,'Position',[8 90 792 420]);  % 生成空的坐标窗口
axis square;hold on;                         % 设置坐标轴属性
[x,y]=meshgrid(0:N);                         % 生成格点的坐标值
plot(N/2,N/2,'r.','markersize',20);          % 画出中心点
set(gca,'Position',[0.05 0.11 0.775 0.815]); % 设置坐标轴大小
k=1/3;ks=sqrt(k)*0.7;                        % 设置比例系数
r=N*k;                                     % 鱼眼透镜半径的大小
b=r^2;                                     % 得到半径的平方
axis([0,N,0,N]);                             % 设置坐标轴范围
rxy=sqrt([x-N/2].^2+[y-N/2].^2);             % 计算网格点到中心的距离
for m=1:N+1;                                % 逐点计算
   for n=1:N+1;
      if rxy(m,n)>1e-3 & rxy(m,n)<r;
         A=angle([x(m,n)-N/2]+i*[y(m,n)-N/2]); % 计算当前格点对中心的方位角度
         a=2*r^2/rxy(m,n);                   % 计算参数 a
         rr=-1/2*a+1/2*(a^2+4*b)^(1/2);        % 得出目标点在鱼眼镜头成像下到中心的距
离
         x(m,n)=rr*cos(A)+N/2;               % 根据角度和到中心的距离得到像点坐标
         y(m,n)=rr*sin(A)+N/2;               % 根据角度和到中心的距离得到像点坐标
      else
         x(m,n)=[x(m,n)-N/2]*ks+N/2;          % 处理透镜半径之外点坐标的计算
         y(m,n)=[y(m,n)-N/2]*ks+N/2;     % 处理透镜半径之外点坐标的计算
      end
```

```
    end
end
for k=1:N+1;
    plot(x(:,k),y(:,k),'k');  % 根据数据点绘图
    plot(x(k,:),y(k,:),'k');  % 根据数据点绘图
end
axis image;              % 设置坐标轴属性
set(gca,'xtick',[],'ytick',[]);  % 删去坐标轴刻度
title(['fish-eye']);         % 加注图题
```

首先对相关的几何参数赋值，然后对鱼眼透镜内部区域和外部区域采用不同计算方法。遍历网格矩阵的所有交点后，在 X 轴方向和 Y 轴方向分别利用 plot 函数绘制网状图，然后就可以得到相应的鱼眼效果图。上述程序执行后可以得到如图 24.3 所示的图形。

如果把中心点随机地移动并实时计算当前鱼眼效果图，可以得到一个动画过程。相应地实现程序为光盘中\Ch24 文件夹下的 fish_eye2.m 文件。其中一个截图如图 24.4 所示。

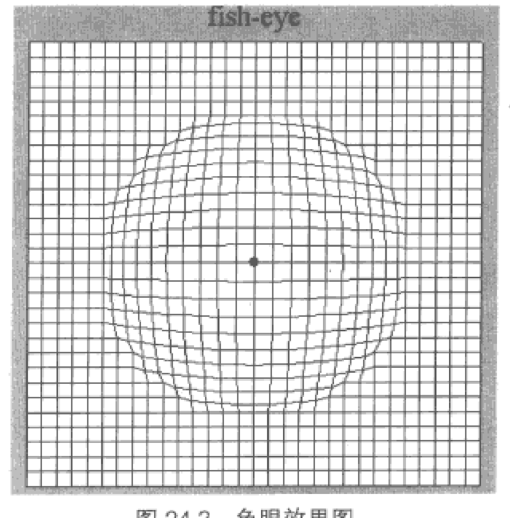



图 24.3　鱼眼效果图

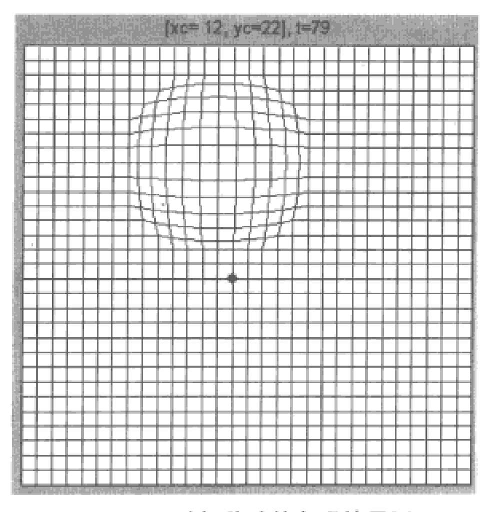


图 24.4　随机移动的鱼眼效果图

在前面给出鱼眼效果实现的程序中，两个参数 k 和 ks 之间的关系式是 ks=sqrt(k)*0.7，这是一个经验公式。其中的系数“0.7”是随着 k 的变化而变化的，这里把这个关系式总结为：

$$ks = c\sqrt{k} \tag{24-5}$$

通过测试，不同的 k 对应的 c 数值如表 24.1 所示。

表 24.1　参数 k 和 c 的部分经验数值

k	c	k	c	k	c	k	c
0.20	0.92	0.30	0.75	0.40	0.65	0.50	0.58
0.23	0.86	0.32	0.73	0.42	0.635		
0.25	0.83	0.35	0.70	0.45	0.62		
0.27	0.80	0.37	0.68	0.47	0.61		

可以利用下面一段小程序来计算[0.2,0.5]之间的 k 值对应的 c 数值。

```
function c=fitfc(k);
```

```
kn=[0.2,0.23,0.25,0.27,0.3,0.32,0.35,0.37,0.4,0.42,0.45,0.47,0.5];     % 离散的数据
cn=[0.92,0.86,0.83,0.8,0.75,0.73,0.7,0.68,0.65,0.635,0.62,0.61,0.58];  % 离散的数据
Dd=abs(kn-k);            % 求出输入 k 和经验数据点的差值
[Dm,Ik1]=min(Dd);        % 找出最小值
Dd(Ik1)=3;               % 把当前最小值变为一个较大的数
[Dm,Ik2]=min(Dd);        % 找出当前最小值，这样就得到了两个比较小的数值，它们包含了输入点
c=cn(Ik1)*[k-kn(Ik1)]/[kn(Ik2)-kn(Ik1)]+cn(Ik2)*[kn(Ik2)-k]/[kn(Ik2)-kn(Ik1)];
% 用最近两点拟合
```

同时把前面绘制鱼眼的程序改为函数文件的形式，其中 N 和 k 作为函数的输入参数，而上面的函数 fitfc 作为子函数，这样可以得到 draw_fisheye.m 文件，该文件保存在光盘中\Ch24 文件夹下。通过下面的语句可以得到如图 24.5 所示的图形对话窗口。

```
axes('position',[0.1,0.1,0.6,0.6]); % 设置坐标轴的位置
draw_fisheye(30,0.20); % 绘制默认值时的鱼眼效果图
sl=uicontrol(gcf,'style','slider','unit','normalized','position',[0.84,0.1,0.04,0.8],...

'TooltipString','k=0','SliderStep',[0.01,0.01],'Callback',['v=get(sl,''value'');cla;',...

'draw_fisheye(30,0.2+0.4*v);set(sl,''TooltipString'',[''k='',num2str(0.2+0.4*v)])']); % 生成滑动条控件
```

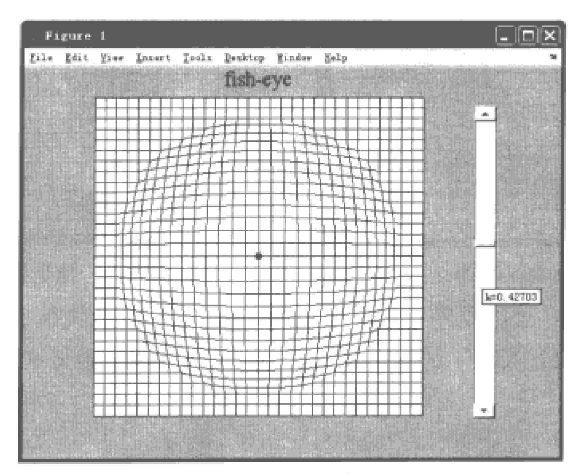


图 24.5　带有滑动条改变参数的 GUI 窗口

24.2　计算全息编码及再现程序

全息术是在光学衍射理论和激光器的基础上建立起来的一种光学技术，它可以用于信息处理领域中的信息存储和损伤检测等问题。计算全息是建立在光学衍射理论和光学全息基础上的一种离散形式的理论，是光谱学、物理光学、计算物理以及计算机科学等多门学科的综合。而全息编码技术是计算全息中的一个重要环节。

计算全息编码是把复振幅信号转换为实数、非负的函数值，以便于在输出介质上显示。记录了

全息图的信息介质，可以通过光学再现系统来再现全息图像。目前存在的编码技术可以分为三类。第一类是对振幅和相位进行编码，其特点是振幅和相位完全是非负的实数而无须考虑负数的问题。在这种编码过程中，一般是使用矩形孔的面积、形状以及位置进行调制。这种编码技术要求绘制编码图案时的定位精度高。第二种编码方法是增加离轴的参考光波，从而增加偏置项，直接对振幅进行编码。这种编码方法需要与参考光波进行叠加运算，因此其计算量相对大一些。而且再现的时候需要引入相同的参考光波进行照明。第三种编码方法是对振幅进行正交分解，其中比较典型的是四级迂回式相位编码技术。可见直接对复振幅的虚部和实部进行编码时计算量较小。

下面来介绍迂回相位编码的具体方法，这个编码方法最早是由德国科学家 Lohmann 提出的，其原理如图 24.6 所示。图中的正方形虚线被认为是一个编码单元，其中实线围成的矩形框表示这个编码单元内的“透光部分”(对应的元素值等于 1)。而在这个编码单元内矩形框之外的部分认为是不透光的，即对应的元素等于 0。矩形的高等于 $2h$，这个数值对应于离散单元在傅里叶变换谱中的振幅大小，矩形的宽度取固定的数值。而这个矩阵的中心轴距 Y 轴的距离 k_z 对应于变换谱的相位值，其位于 Y 轴的左右两侧分别对应负和正的位相。这种编码方式被称为罗曼Ⅲ型编码。

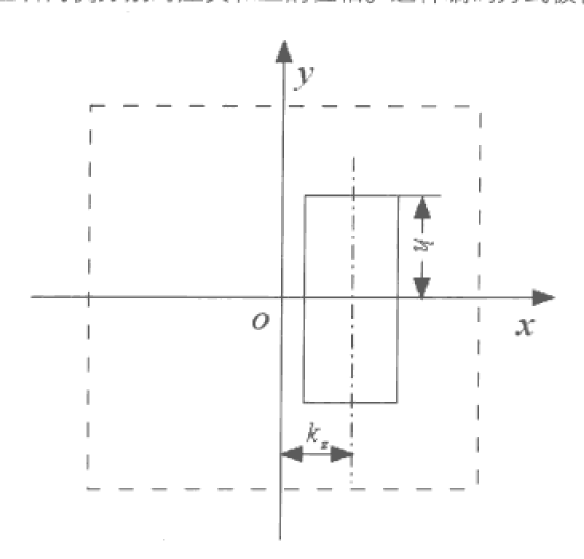


图 24.6 迂回相位编码计算的原理

这里使用一个 11×11 的矩阵对应于图 24.6 所示的编码单元。相应的计算全息编码和再现的 MATLAB 程序如下：

```
A=zeros(64);      % 生成全 0 矩阵
A(15:20,20:40)=1; % 生成字母 E 的部分线条
A(15:50,20:25)=1; % 生成字母 E 的部分线条
A(45:50,20:40)=1; % 生成字母 E 的部分线条
A(30:34,20:35)=1; % 生成字母 E 的部分线条
figure;imshow(1-abs(A),[]); % 显示图像并进行反色处理
Fa=fft2(fftshift(A)); % 进行傅里叶变换
Fs=fftshift(Fa);     % 进行频移操作
Am=abs(Fs);         % 取模
Ph=angle(Fs);        % 取相位
s=11;              % 设置编码单元的大小
cgh=zeros(64*s);    % 生成全 0 计算全息编码矩阵
th=max(max(abs(Fs))); % 计算出频谱中最大的振幅值
```

```
qq=th/1.2;            % 对最大值进行截断
Am(Am>qq)=qq;       % 把阈值以上部分的数值设置为阈值
q=1:s;              % 生成单元格中的离散序列
w=(s+1)/2;           % 计算出中心坐标
for m=1:64;          % 遍历各行
    for n=1:64;       % 遍历各列
        h=round(Am(m,n)/qq*(w-1)-0.5);  % 把振幅进行归一化处理并进行量化
        md=zeros(s);  % 初始化单元格内的数值
        if h>0;        % 对正数的振幅进行编码
            td=ones(h*2+1,3);          % 生成矩形透明孔
            Pm=round(Ph(m,n)/pi*3);  % 对相位进行编码
            kz=Pm+w;                 % 计算透明孔的列位置
            md(w-h:w+h,kz-1:kz+1)=td; % 按列位置进行赋值
        end
        cgh((m-1)*s+q,(n-1)*s+q)=md;  % 把计算生成的单元孔放置到对应的位置处
    end
end
figure;imshow(max(max(cgh))-cgh,[]);     % 迂回位相编码结果
Re=ifft2(cgh);                          % 利用逆傅里叶变换进行再现
Re=fftshift(Re);                         % 进行频谱移动
figure;imshow(max(max(abs(Re)))-abs(Re),[]);  % 再现图像
```

首先对矩阵 $\boldsymbol{A}$ 的各个部分进行复制而得到一个字母“E”图案，然后对矩阵 $\boldsymbol{A}$ 进行傅里叶变换。对变换谱进行全息编码，编码单元是一个 11×11 的小矩阵，并显示编码后的矩阵。最后利用逆傅里叶变换进行全息再现的模拟。执行上述程序，得到的输出图形中使用的输入图像如图 24.7 所示。相应的全息编码图像如图 24.8 所示。再现图像如图 24.9 所示。

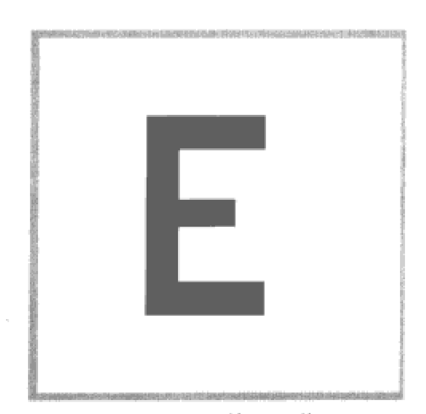


图 24.7　输入图像

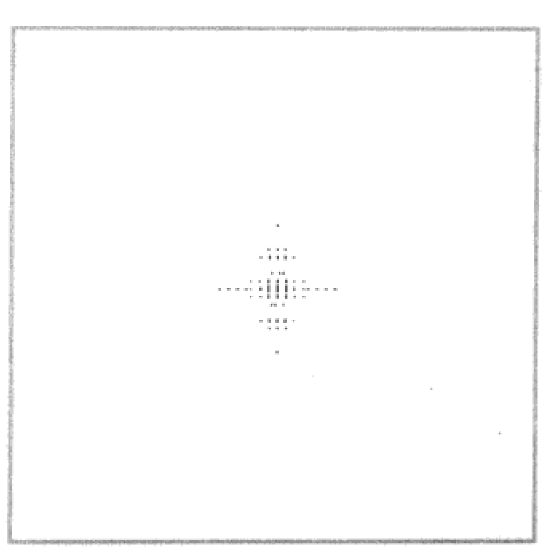


图 24.8　全息迂回相位编码图像

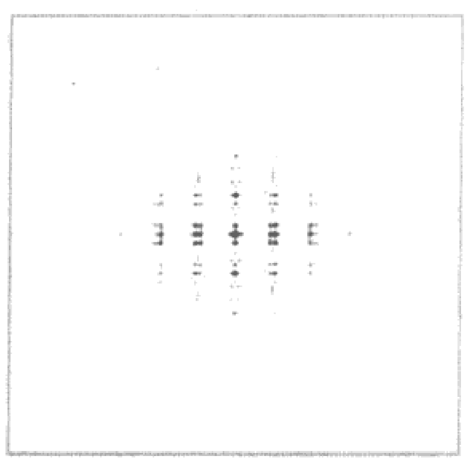


图 24.9 再现图像



图 24.7、图 24.8、图 24.9 使用了反色显示，即黑色对应的数值较大，而白色对应的像素值较小。这样做是为了避免出现大面积的黑色区域。

从图 24.9 可以看出再现图像不够清晰，为了得到更清晰的图像，应该使用元素更多的子矩阵来表示编码单元。相应地对计算机的内存要求更高。此外在编码形式上还可以选择矩形高度固定，利用矩形的宽度表示振幅，利用中心轴在 X 轴上的投影表示相位值。

前面介绍了罗曼Ⅲ型编码，这里再补充罗曼Ⅰ型和罗曼Ⅱ型编码。罗曼Ⅰ型编码方案是矩形孔的高度不变，使用矩形孔的宽度 D 对应于变换谱的振幅值，而使用矩形孔的竖直对称轴对 Y 轴的位移 s 对应于复振幅的相位值，相应的关系图如图 24.10 所示。

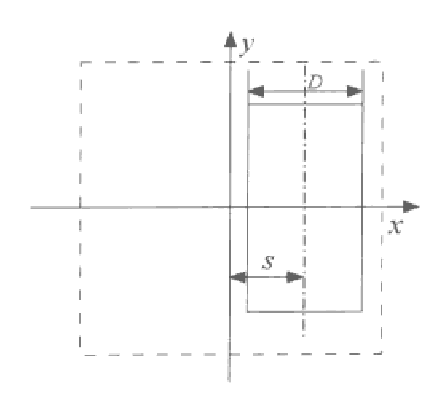


图 24.10 罗曼Ⅰ型编码原理图

在罗曼Ⅱ编码方式中，使用矩形的宽度调制振幅和相位，每个编码单元含有两个宽度和高度相同的矩形孔，它们的间距 D 对应着振幅数值。它们的中心轴到编码单元中心位移 s 对应于相位值。该编码方式的几何关系如图 24.11 所示。

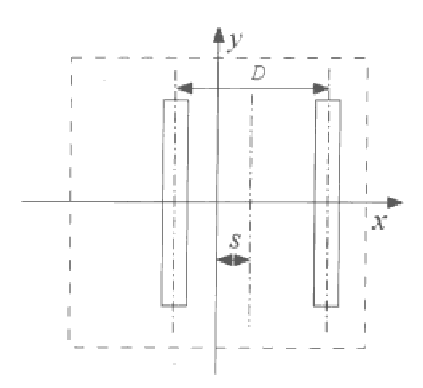


图 24.11 罗曼 II 型编码原理图

上面的罗曼 I 型编码方式和罗曼 II 型编码方式可以模仿前面的罗曼 III 型编码方式建立实现程序。其中罗曼 I 型编码要求的绘图时间短，同时可以在 Y 轴方向抑制多级衍射，然而这种编码方式的精度较低。罗曼 II 型编码同样具有较低的精度，同时绘图时间长，目前应用较少。罗曼 III 型编码吸取了罗曼 I 型编码和罗曼 II 型编码的优点，故其应用广泛。

下面给出博奇（Burch）编码的定义，这种编码方式通过增加直流偏置量来构成全息函数，即：

$$H(u,v)=\frac{1}{2}\left\{1+A(u,v)\left[\cos\left(2\pi ua-\phi(u,v)\right)\right]\right\} \tag{24-6}$$

其中 $H(u,v)$ 表示全息面的谱分布函数。$A(u,v)$ 表示全息波前的振幅，其中 $A(u,v)$ 已经做了归一化处理，最大值为 1。这样通过公式（24-6）的转化，函数 $H(u,v)$ 可以保证各处函数值非负。a 是载频系数。$\phi(u,v)$ 是全息波前的相位函数。式（24-6）中对振幅函数进行了归一化处理，此外还可以取 $A(u,v)$ 的最大值来保证函数 $A(u,v)$ 非负，即：

$$H(u,v)=\frac{1}{2}\left\{\max\left[A(u,v)\right]+A(u,v)\left[\cos\left(2\pi ua-\phi(u,v)\right)\right]\right\} \tag{24-7}$$

其中 max 表示计算二元函数的最大值。在式（24-7）中就不必进行归一化操作了。在编码的时候，利用正方形开口表示光强，正方形的边长对应于 $\sqrt{H(u,v)}$。

24.3 光的等厚干涉

光的等厚干涉现象如图 24.12 所示。一束平行光竖直向下照射到一个倾斜的空气劈尖，在平面上将形成等间距明暗相间的干涉条纹。设相邻暗条纹之间的距离为 l，那么有如下关系成立：

$$l\sin\alpha=\lambda/2 \tag{24-8}$$

其中 α 为空气劈尖的顶角大小。λ 为入射光波的波长。这样如果已知光波波长，同时测量出条纹之间的距离就可以得出两个平面之间的夹角。

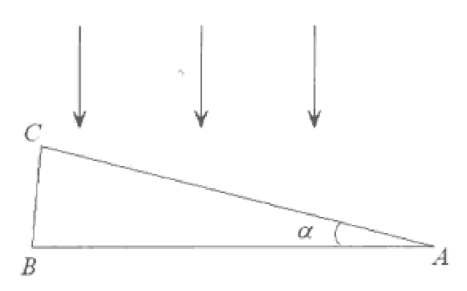


图 24.12 空气劈尖等厚干涉原理图

下面给出空气劈尖等厚干涉的动画演示程序。

```
set(gcf,'DoubleBuffer','on'); % 设置渲染效果
axes('position',[0.12,0.08,0.6,0.8]);hold on; axis manual; % 设置坐标轴属性
title('关于光的等厚干涉(劈尖)的动画','fontsize',16);      % 添加图题
rectangle('position',[0.15,0.2,0.7,0.01],'FaceColor',[0.1,0.3,0.4]);% 画一个矩
形，作为底面
plot([0.2,0.2],[0.2,0.4]); % 画一条竖线
plot([0.8,0.2],[0.2,0.4]); % 绘制斜面
style='none'; % 擦除模式
for k=1:10;
    h(k)=plot([0.21+(k-1)*0.06]*[1,1],[0.6,0.6],'r',...
        'linewidth',1,'EraseMode',style); %画出多条入射光线
end
G=0.6;        % 设置入射光线随时间变化的下端点位置
while G>=0.2; % 循环计算
   G=G-0.01; % 计算当前时刻的下端点位置
   set(h,'ydata',[0.6,G]); % 更新绘图数据
   pause(0.1);          % 暂停一下显示动画效果
end
for k=1:10;
    hr(k)=plot([0.21+(k-1)*0.06]*[1,1],[0.2,0.2],'r',...
        'linewidth',2); % 设置底面反射光线的起点
end
while G<0.4;   % 循环绘制反射光线
   G=G+0.01; % 计算当前反射光线的位置
   set(hr,'ydata',[0.2,G]); % 更新反射光线的位置
   pause(0.1);        % 暂停一下显示动画效果
end
z=[0.8,0.2]+i*[0.2,0.4]; % 生成沿着斜面的向量
N=400; % 计算采样点数
An=angle(diff(z))-pi/2; % 计算角度值
An=exp(i*An);         % 模值为 1 的复数
zz=linspace(z(1),z(2),N); % 离散取样点值
Si=abs(sin(linspace(0,pi*20,N))).^2/20; % 生成干涉曲线
zz=zz+Si*An; % 进行旋转和平移操作
plot(zz,'b');    % 绘制干涉曲线
```

首先利用函数 plot 生成一个劈尖图案。然后模拟光入射过程的动画，光线每步增加一定的长度，到达劈尖底部时光线将发生反射，在劈尖倾斜的面上发生干涉现象，得到一系列的干涉条纹。这个程序可以演示光线入射和反射的过程，执行程序可以得到如图 24.13 所示的图形。

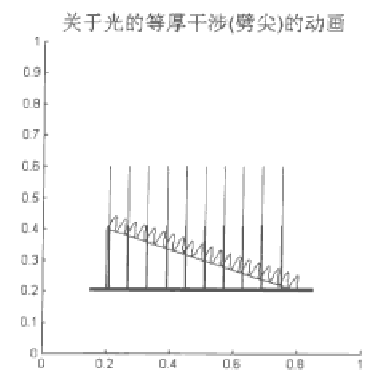


图 24.13 劈尖等厚干涉模拟结束后的图案

24.4 杨氏双缝干涉

杨氏双缝干涉实验是光学中的一个重要实验，它在量子理论中也是一个重要的模型。英国物理学家托马斯·杨最先在 1801 年得到了两列相干的光波，同时以明确的形式建立了光波叠加原理，用光的波动性成功地解释了光学干涉现象。此外，他用较高强度的单色光垂直照射到开有小孔 S 的不透明遮光板（其在光学中也称为光阑）上，如图 24.14 所示。在 P_2 后面放置第二块光阑，上面有两个小孔 S_1 和 S_2。在 P_3 平面上可以得到一系列明暗相间的干涉条纹。杨氏利用了惠更斯对光的传播所提出的次波原理解释了这个著名的光学实验。

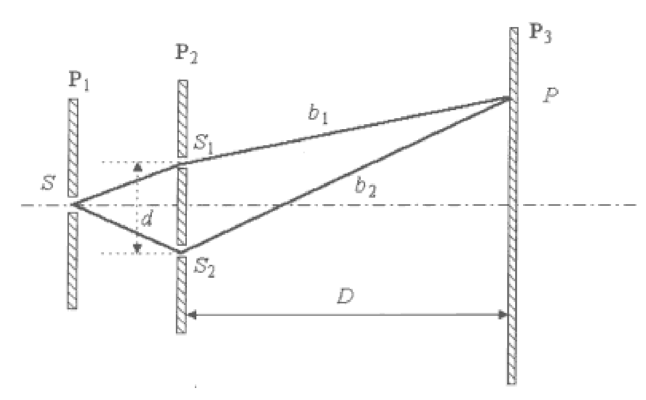


图 24.14 杨氏双缝干涉原理图

编写程序模拟杨氏双缝干涉的过程，并给出杨氏双缝干涉的演示程序。可以按照图 24.15 所示的流程编写模拟程序。

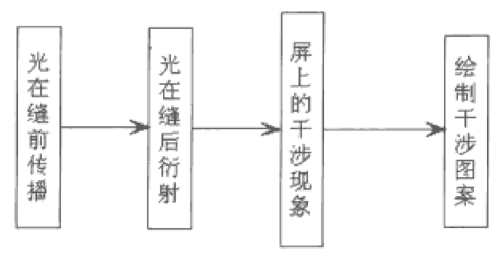


图 24.15 杨氏双缝干涉模拟的流程图

```
figure('position',[217 266 694 244]); % 设置图形窗口的位置属性
set(gcf,'DoubleBuffer','on'); % 设置渲染效果
axes('position',[0.12,0.08,0.6,0.8]);hold on; % 设置坐标轴位置
rectangle('position',[-1,1.1,0.1,3],'FaceColor',[0.1,0.3,0.4]);    % 绘制矩形作为
光阑的一部分
rectangle('position',[-1,-4,0.1,3],'FaceColor',[0.1,0.3,0.4]);     % 绘制矩形作为
光阑的一部分
rectangle('position',[-1,-0.8,0.1,1.6],'FaceColor',[0.1,0.3,0.4]);  % 绘制矩形作
为光阑的一部分
axis([-4,12,-5,5]); % 设置坐标轴范围
ha=plot([-3.8,-3.8],[0.9,0.9],'r','linewidth',3); % 绘制其中一条入射光线
hb=plot([-3.8,-3.8],[-0.9,-0.9],'r','linewidth',3); % 绘制其中第二条入射光线
title('Double slit interference  \copyright xxx'); % 加注图题
L=-3.8; % 入射光线的设置端点位置
rectangle('position',[11,-4,0.1,8],'FaceColor',[0,0,0]); % 绘制后面的屏
plot([-1,4.1],[-3.5,-3.5],'k'); % 增加长度标注的水平线段部分
plot([5.9,11],[-3.5,-3.5],'k'); % 增加长度标注的水平线段部分
plot([-0.7,-1,-0.7],[-3.3,-3.5,-3.7],'k'); % 画箭头中的短线
plot([10.7,11,10.7],[-3.3,-3.5,-3.7],'k'); % 画箭头中的短线
text(4.8,-3.5,'3m'); % 添加标注长度的文字
plot([-1.5,-1.5],[0.9,0.35],'k');  % 增加长度标注的竖直线段部分
plot([-1.5,-1.5],[-0.9,-0.35],'k'); % 增加长度标注的竖直线段部分
plot([-1.65,-1.5,-1.3],[0.6,0.9,0.6],'k'); % 画箭头中的短线
plot([-1.6,-1.5,-1.3],-[0.6,0.9,0.6],'k'); % 画箭头中的短线
text(-2.1,0,'2mm'); % 加注文字
while L<=-0.95;   % 绘制光在狭缝前的传播情况
   L=L+0.05; % 更新位置
   set(ha,'xdata',[-3.8,L]); % 更新上面一条光线的位置数据
   set(hb,'xdata',[-3.8,L]); % 更新下面一条光线的位置数据
   pause(0.05); % 暂停一下，显示动画效果
end
xas=-0.95;yas=0.9;  % 生成上面小孔发出的光线
xbs=-0.95;ybs=-0.9; % 生成下面小孔发出的光线
po=[-3.5:3.5]; % 离散化数据
ka=(po-yas)*i+(11+0.95);ka=ka./abs(ka); % 生成复数数值
kb=(po-ybs)*i+(11+0.95);kb=kb./abs(kb); % 生成复数数值
for n=1:8; % 循环画衍射光线
   ah(n)=plot(xas,yas,'r'); % 画上面小孔发出的衍射光线
   bh(n)=plot(xbs,ybs,'r'); % 画下面小孔发出的衍射光线
end
r=0; % 长度参数
za=xas+yas*i; % 生成上面小孔的坐标值
zb=xbs+ybs*i; % 生成下面小孔的坐标值
Le=sqrt(11.95^2+3.5^2); % 计算两个小孔中点距离衍射最远点的长度
while r<=Le; % 控制循环终止条件
   for n=1:8; % 循环画线
       Ta=za+r*ka(n); % 计算上面小孔衍射光线坐标点对应的复数
       Ya=imag(Ta); % 计算衍射光线右侧端点的纵坐标
       if abs(imag(Ta))>=abs(po(n)); % 检测光线是否传播到屏所在平面
          Ya=po(n); % 获取屏上的坐标值
       end
       set(ah(n),'xdata',[xas,min(11,real(Ta))],'ydata',[yas,Ya]); % 更新衍射光线
数据
       Tb=zb+r*kb(n); % 计算下面小孔衍射光线坐标点对应的复数
       Yb=imag(Tb);  % 计算衍射光线右侧端点的纵坐标
```

```
        if abs(imag(Tb))>=abs(po(n));  % 检测光线是否传播到屏所在平面
            Yb=po(n); % 获取屏上的坐标值
        end
        set(bh(n),'xdata',[xbs,min(11,real(Tb))],'ydata',[ybs,Yb]); % 更新衍射光线
数据
    end
    r=r+0.05; % 更新光线传播距离的数值
    pause(0.05);
end
s=meshgrid(linspace(4,-4,300))'; % 生成坐标矩阵
De=abs((s-1)*i+3000)-abs((s+1)*i+3000); % 计算波程差
lambda=0.6328e-3;   % 对波长赋值
de_A=De/lambda*pi*2; % 计算波程差
It=1-cos(de_A); % 计算干涉图案
axes('position',[0.72,0.16,0.26,0.64]);     % 生成一个新的坐标轴
cc=cat(3,It/2,zeros(size(It)),zeros(size(It))); % 生成彩色图案对应的数据矩阵
imshow(cc,[]); % 显示干涉图案
xlabel('\copyright MATLAB 2008'); % 对图形添加标注
```

首先模拟光在入射到小孔前的传播情况模拟。到达小孔光将发生衍射，向不同方向传播。这时一根光线将分为多根光线，同时它们向屏所在平面传播。到达屏面后将发射干涉产生明暗条纹，同时给出干涉图案的显示。运行上述程序可以得到杨氏双缝干涉实验的模拟，程序运行完毕后可以得到如图 24.16 所示的图形。两束光首先平行于水平轴向右传播，遇到小孔后发生衍射，向不同方向传播。当它们传播到后面竖直放置的接收屏时两束发散的光束进行干涉，这时可以得到如图 24.16 中右侧图形所示的干涉图案。这个程序形象地模拟了杨氏双缝干涉现象的动态过程。



这里使用函数 rectangle 来绘制光阑，读者还可以使用函数 plot 或者 line 绘制一条粗的实线来表示光阑，从而简化程序。

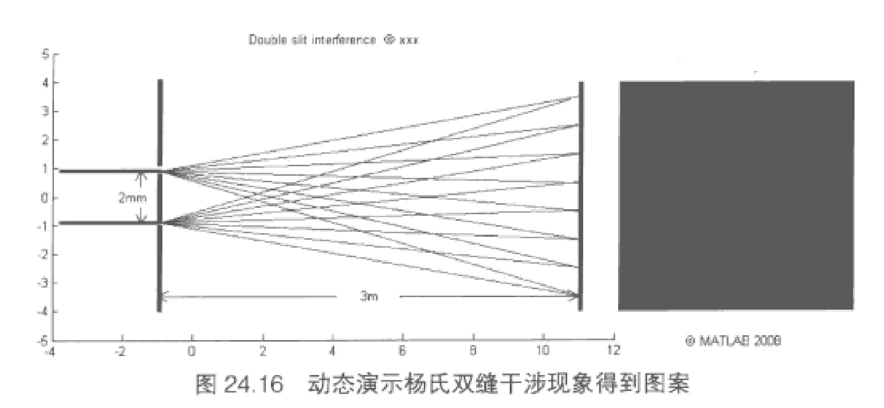


图 24.16　动态演示杨氏双缝干涉现象得到图案

24.5　牛顿环

如果把一块曲率半径 R 很大的平凸透镜的凸面放在一块光学平板玻璃上面（如图 24.17 所示），这样就在透镜的凸面和平板玻璃间形成一个空气薄层，其厚度从中心接触点到边缘逐渐增加。而离接触点（或者说平凸透镜最低点）等距离的地方，厚度相同，等厚膜的轨迹是以接触点为中心的圆。当波长为 λ 的单色光垂直入射透镜时，在空气膜上下表面反射的两束光为相干光束，从而形成的干

涉条纹是以透镜与玻璃接触点为圆心的明暗相间的同心圆环，这些圆环被称为牛顿环。

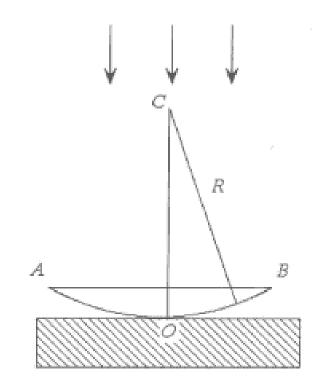


图 24.17 牛顿环的原理图

下面来模拟牛顿环图案，这里给出一个用户交互操作界面的实现程序，其中包括：

- 牛顿环原理图的绘制。
- 牛顿环图案的模拟。
- 控制参数的调节。

相应程序内容如下：

```
figure('Position',[90 164 873 483]); % 设置图形窗口位置
L=632.8;R=5;H=0.005; % 设置默认参数值：波长 L、半径 R 和厚度 H
a1=axes('Position',[0.83,0.3,0.15,0.4]); % 设置坐标轴位置，在其上绘制简易的原理图
hold on;axis([0,1,0,1]); % 设置坐标轴范围
plot([0.25,0.25],[0.5,0.8],'k'); % 画竖直线，入射光线
plot([0.5,0.5],[0.5,0.8],'k');   % 画竖直线，入射光线
plot([0.75,0.75],[0.5,0.8],'k'); % 画竖直线，入射光线
fill([0.22,0.25,0.27],[0.5,0.44,0.5],'k'); % 画出箭头
fill([0.47,0.5,0.52],[0.5,0.44,0.5],'k');  % 画出箭头
fill([0.72,0.75,0.77],[0.5,0.44,0.5],'k'); % 画出箭头
z=1.8*exp(i*(linspace(-0.21,0.21,30)-pi/2))+2.1*i+0.5; % 生成平凸透镜的轮廓
fill(real(z),imag(z),'w'); % 利用 fill 函数画平凸透镜的轮廓
rectangle('Position',[0.1,0.18,0.8,0.12]); % 画玻璃板底面
set(gca,'xtick',[],'ytick',[],'box','on');     % 删去坐标轴刻度
a2=axes('Position',[0.4,0.16,0.4,0.7]);  % 生成第二个坐标轴，在其上画牛顿环
[x,y]=meshgrid(linspace(-0.005,0.005,200)); % 生成坐标矩阵
r2=(x.^2+y.^2); % 坐标矩阵各点到中心的距离
Di=[2*H+2*(R-sqrt(R^2-r2))*1e9]/L; % 计算光程差
In=abs(cos(Di*pi*2)); % 得到牛顿环上的光强数值
cr=abs(L-560)/200;  % 红色的分量值
cg=1-cr; % 绿色的分量值
cb=abs(L-600)/240; % 蓝色的分量值
Ik(:,:,1)=In*cr; % 转化为 RGB 三基色表示颜色矩阵数值（红色）
Ik(:,:,2)=In*cg; % 转化为 RGB 三基色表示颜色矩阵数值（绿色）
Ik(:,:,3)=In*cb; % 转化为 RGB 三基色表示颜色矩阵数值（蓝色）
Pc=imshow(Ik,[]); % 绘制默认参数下的数值
title('the pattern of Newton''s rings','fontsize',18); % 添加图题
Lt=uicontrol(gcf,'style','text',...
 'unit','normalized','position',[0.06,0.86,0.21,0.06],...
```

```
 'BackgroundColor',0.7*[1,1,1],'ForegroundColor',[0.8,0.1,0.9],...
 'string','wavelength: 632.8nm','fontsize',16,'fontname','times new roman'); % 实时显示波长的文本框
s1=uicontrol(gcf,'style','slider',...
  'unit','normalized','position',[0.06,0.76,0.21,0.04],...
  'BackgroundColor',0.7*[1,1,1],'ForegroundColor',[0.1,0.1,0.9],...
  'SliderStep',[0.01,0.01],'value',(632.8-360)/400,...
  'callback',['L=get(s1,''value'')*400+360;',...
  'set(Lt,''string'',[''wavelength: '',num2str(L/10),''nm'']);',...
  'Di=[2*H+2*(R-sqrt(R^2-r2))*1e9]/L;',...
  'In=abs(cos(Di*pi*2));cr=abs(L-560)/200;cg=1-cr;',...
  'cb=abs(L-600)/240;Ik(:,:,1)=In*cr;Ik(:,:,2)=In*cg;',...
  'Ik(:,:,3)=In*cb;set(Pc,''CData'',Ik);']); % 利用滑动条改变波长数值
uicontrol(gcf,'style','text',...
  'unit','normalized','position',[0.04,0.81,0.08,0.04],...
  'BackgroundColor',0.8*[1,1,1],'ForegroundColor',[0.1,0.1,0.9],...
  'string','360','fontsize',16,'fontname','times new roman'); % 显示最小波长数值
uicontrol(gcf,'style','text',...
  'unit','normalized','position',[0.22,0.81,0.08,0.04],...
  'BackgroundColor',0.8*[1,1,1],'ForegroundColor',[0.1,0.1,0.9],...
  'string','760','fontsize',16,'fontname','times new roman'); % 显示最大波长数值
Rt=uicontrol(gcf,'style','text',...
  'unit','normalized','position',[0.06,0.66,0.23,0.06],...
  'BackgroundColor',0.7*[1,1,1],'ForegroundColor',[0.8,0.1,0.9],...
  'string','radii:','fontsize',16,'fontname','times new roman'); % 实时显示半径数值的文本框
s2=uicontrol(gcf,'style','slider',...
  'unit','normalized','position',[0.06,0.56,0.21,0.04],...
  'BackgroundColor',0.7*[1,1,1],'ForegroundColor',[0.1,0.1,0.9],...
  'SliderStep',[0.01,0.01],...
  'callback',['R=get(s2,''value'')*7+5;',...
  'set(Rt,''string'',[''radii: 5m'',num2str(R),''m'']);',...
  'Di=[2*H+2*(R-sqrt(R^2-r2))*1e9]/L;',...
  'In=abs(cos(Di*pi*2));cr=abs(L-560)/200;cg=1-cr;',...
  'cb=abs(L-600)/240;Ik(:,:,1)=In*cr;Ik(:,:,2)=In*cg;',...
  'Ik(:,:,3)=In*cb;set(Pc,''CData'',Ik);']); % 通过滑动条改变半径数值
uicontrol(gcf,'style','text',...
  'unit','normalized','position',[0.04,0.61,0.08,0.04],...
  'BackgroundColor',0.8*[1,1,1],'ForegroundColor',[0.1,0.1,0.9],...
  'string','5','fontsize',16,'fontname','times new roman'); % 显示最小半径
uicontrol(gcf,'style','text',...
  'unit','normalized','position',[0.22,0.61,0.08,0.04],...
  'BackgroundColor',0.8*[1,1,1],'ForegroundColor',[0.1,0.1,0.9],...
  'string','12','fontsize',16,'fontname','times new roman'); % 显示最大半径
Ht=uicontrol(gcf,'style','text',...
  'unit','normalized','position',[0.06,0.46,0.23,0.06],...
  'BackgroundColor',0.7*[1,1,1],'ForegroundColor',[0.8,0.1,0.9],...
  'string','thickness: 5nm','fontsize',16,'fontname','times new roman'); % 实时显示厚度的文本框
s3=uicontrol(gcf,'style','slider',...
  'unit','normalized','position',[0.06,0.36,0.21,0.04],...
  'BackgroundColor',0.7*[1,1,1],'ForegroundColor',[0.1,0.1,0.9],...
  'SliderStep',[0.01,0.01],'value',0.05,...
  'callback',['H=get(s3,''value'')*0.01;',...
  'set(Ht,''string'',[''thickness: '',num2str(H),''nm'']);',...
  'Di=[2*H+2*(R-sqrt(R^2-r2))*1e9]/L;',...
```

```
  'In=abs(cos(Di*pi*2));cr=abs(L-560)/200;cg=1-cr;',...
  'cb=abs(L-600)/240;Ik(:,:,1)=In*cr;Ik(:,:,2)=In*cg;',...
  'Ik(:,:,3)=In*cb;set(Pc,''CData'',Ik);']); % 通过滑动条改变厚度数值
uicontrol(gcf,'style','text',...
  'unit','normalized','position',[0.04,0.41,0.08,0.04],...
  'BackgroundColor',0.8*[1,1,1],'ForegroundColor',[0.1,0.1,0.9],...
  'string','0','fontsize',16,'fontname','times new roman'); % 显示最小厚度
uicontrol(gcf,'style','text',...
  'unit','normalized','position',[0.22,0.41,0.08,0.04],...
  'BackgroundColor',0.8*[1,1,1],'ForegroundColor',[0.1,0.1,0.9],...
  'string','0.01','fontsize',16,'fontname','times new roman'); % 显示最大厚度
```

首先利用函数 axes 在图形窗口右侧画出一个小坐标轴，在其上利用函数 plot 和 fill 得到牛顿环原理图的绘制。然后在图形窗口中间添加一个坐标轴用于模拟牛顿环图案。最后在图形窗口左侧布置调节牛顿环参数的 GUI 控件，并调节波长、透镜表面的曲率半径以及空气隙的厚度。

执行上述程序可以得到如图 24.18 所示的界面形式，通过 3 个滑动条可以改变波长、半径以及厚度 3 个参数的取值。当前的参数取值显示在滑动条的上侧。这里使用 3 个近似的波长与颜色的对应关系，波长变化时相应牛顿环的颜色也会随之改变。

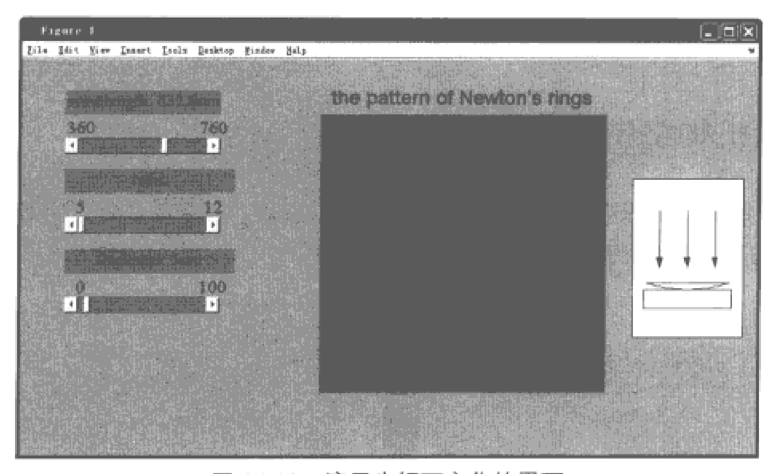


图 24.18　演示牛顿环变化的界面

24.6　小结

光学现象的模拟在当前研究和教学中具有重要的作用，本章给出了几种光学现象的模拟方法。首先介绍了鱼眼效果模拟的方法，给出网格上的鱼眼效果绘制方法。其次介绍了全息编码的实现方法，并给出相应的实现程序。利用动画的形式模拟了劈尖等厚干涉以及杨氏双缝干涉。最后以用户界面形式给出了牛顿环模拟的人机交互控制实例。

◆　◆　◆

第 25 章 机械运动模拟

本章包括

- **凸轮机构绕中轴线旋转** 给出一个凸轮绕其轴线旋转的模拟。
- **阻尼运动** 给出两种阻尼运动的 MATLAB 模拟。
- **连杆机构的运动模拟** 介绍了双摆、四连杆、套环、三弹簧振子模型的程序模拟。
- **凸轮的转动** 给出模拟凸轮带动椭圆面旋转的例子。

物体之间或同一物体各部分之间相对位置随时间的变化叫做机械运动。模拟机械运动需要了解系统各个部分的相互受力情况，进而根据牛顿定律来计算系统各个部分在不同时刻的变化规律。MATLAB 具有数值计算和数据可视化的功能，这样就可以方便地模拟机械运动。本章来介绍利用 MATLAB 模拟几种简单的机械运动形式。通过了解这些实例，读者可以模拟其他相关系统的机械运动问题。

25.1 凸轮机构绕中轴线旋转

在如图 25.1 所示的系统中，凸轮围绕中心轴逆时针旋转。在竖直方向上的滑块可以上下移动，但是其受到重力的作用将始终与凸轮边缘相连接。

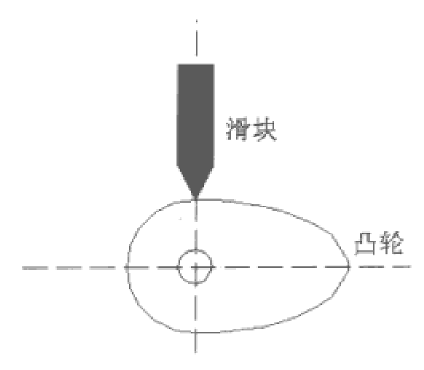


图 25.1 凸轮结构示意图

模拟凸轮结构运动的过程可以分为两部分内容，即：

- 凸轮的旋转运动。
- 滑块的上下运动。

通过对这两部分位置的更新可以得到运动过程的动画显示，相应的实现程序如下：

```
y1=[90 88 82.2 73.7 62 50 37.6 26.7 17.5,...
  11.8 10 10 11 12.4 14 16.8 20 25 31.8,...
```

```
   47.2 50 52.7 68.3 74.9 79.5 83.1 85.6 87.3 88.8 89.7 90]; % 凸轮外面轮廓数据纵坐标
x1=[110 98 86.7 77.8 72 70 72 77.4 86.5 98 110 120,...
      130 140 150 160 169.8 180 189.8 200 200 200 189.9,...
      180 169.9 160 150 140 130 120 110]; % 凸轮外面轮廓数据横坐标
x2=[110 120 120 100 100 110]; % 铅笔状滑块横坐标数据
y2=[90 110 170 170 110 90];  % 铅笔状滑块纵坐标数据
x3=[110 105 101 100 101 105 110 115 118 120 118.6 115 110]; % 凸轮内部的轮廓横坐标数据
y3=[60 58.6 55 50 45 41.3 40 41.3 45 50 55 58.6 60];  % 凸轮内部的轮廓纵坐标数据
x4=[-30,250];   % 水平轴线的横轴数据
x5=[110 110];   % 竖直轴线的横轴数据
y4=[50 50];     % 水平轴线的纵轴数据
y5=[-40,240];   % 竖直轴线的纵轴数据
H1=plot(x1,y1,'r');   % 绘制凸轮外面轮廓
hold on;
H2=fill(x2,y2,'r');     % 绘制铅笔状滑块
H3=plot(x3,y3,'b');   % 绘制凸轮内部轮廓
plot(x4,y4,'k--',x5,y5,'k--'); % 绘制水平和竖直方向的轴线
axis([-30,250,-40,240]);axis square; % 设置坐标轴范围和属性
set(gcf,'doublebuffer','on'); % 设置图形窗口的渲染效果
xlabel('Please press "space" key and stop this program!',...
  'fontsize',14,'color','r'); % 添加提示字符串
k=1; % 控制循环是否执行的参数
t=0; % 记录时间的参数
xy0=[110*ones(31,1),50*ones(31,1)]; % 生成位置数据
while k; % 循环过程模拟凸轮旋转的过程模拟
  s=get(gcf,'currentkey'); % 获得当前键入的按键名称
  if strcmp(s,'space'); % 检查是否为空格键
     clc;k=0; % 清屏并设置参数 k 等于 0，从而可以结束程序
  end
  pause(0.3); % 暂停一下显示动画效果
  t=t+0.1;    % 更新时间
  AA=pi*2*t;  % 计算旋转角度
  Tr=([x1;y1]'-xy0)*[cos(AA),sin(AA);-sin(AA),cos(AA)]; %利用坐标旋转矩阵计算下一时刻的坐标值
  Tr=Tr+xy0; % 进行平移
  X1=Tr(:,1)'; % 获取当前时刻的坐标:横坐标
  Y1=Tr(:,2)'; % 获取当前时刻的坐标:纵坐标
  set(H1,'xdata',X1,'ydata',Y1); % 更新外轮曲线数据
  Tx1=X1;Ty1=Y1; % 对中间变量 Tx1 和 Ty1 赋值
  Tx1(Y1<50)=[];Ty1(Y1<50)=[]; % 删去 Y1 小于 50 的数值
  [Px,K]=sort(abs(Tx1-110));    % 对 Tx1 进行平移和排序
  Py=Ty1(K);     % 对 Py 的顺序进行重排
  if abs(Px(1))<1;
     Yp=Py(1);  % 获取凸轮顶点坐标
  else
     Py=Ty1(K); % 对凸轮坐标排序
     P=polyfit(Px(1:2),Py(1:2),1); % 进行多项式拟合
     Yp=polyval(P,0); % 计算拟合的函数值
  end
  Yh2=y2-90+Yp;    % 计算铅笔状滑块的位置
  set(H2,'ydata',Yh2); % 设置滑块的位置
end
```

首先输入凸轮轮廓结构的数据，并绘出凸轮和滑块的图形，其中把滑块内部填充为红色。接下来计算在不同时刻凸轮的位置，并根据凸轮和滑块的接触点得到滑块的位置，从而可以实时地更新它们的位置。执行上述程序可以得到如图 25.2 所示的图形。这个程序将一直执行下去，如果按下空格键，动画将被停止。

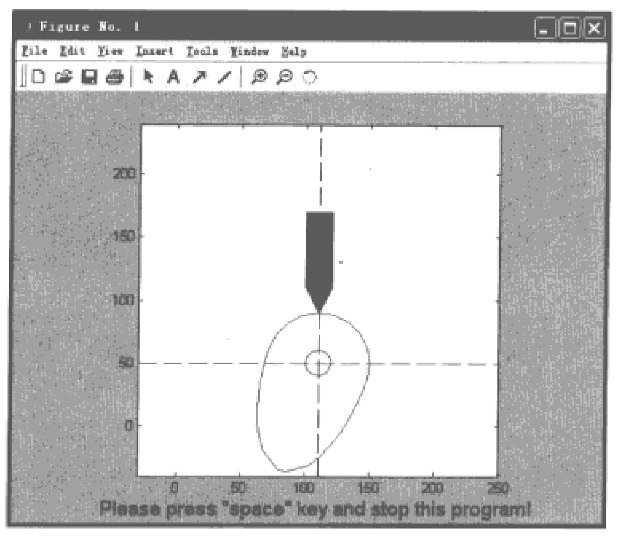


图 25.2　凸轮旋转的动画截图

此外还可以考虑一些特殊函数构成的凸轮轮廓，比如：

$$x^2 + y^2 \mathrm{acos}\left(\frac{x}{x^2+1}\right) = 2 \tag{25-1}$$

利用函数 ezplot 可以画出等式（25-1）对应的曲线，即：

```
ezplot('x^2+y^2*acos(x/(x^2+1))-2',[-2,2]);
```

输出图形的形状如图 25.3 所示。

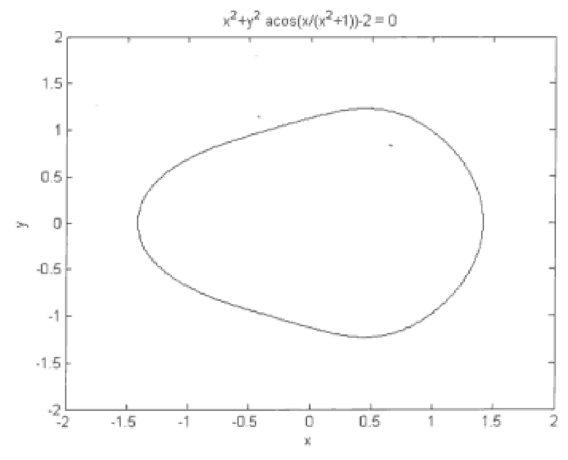


图 25.3　模拟凸轮轮廓的解析图

可以把等式（25-1）改写为下面的形式：

$$y=\pm\sqrt{\left(2-x^2\right)\Big/\operatorname{acos}\left(\frac{x}{x^2+1}\right)} \tag{25-2}$$

这样利用下面的语句即可获取离散的凸轮轮廓数据。

```
t1=linspace(-sqrt(2),sqrt(2),200); % 生成递增数据序列
t2=linspace(sqrt(2),-sqrt(2),200); % 生成递减数据序列
x=[t1,t2];                          % 生成横坐标数据
y=[sqrt((2-t1.^2)./acos(t1./(1+t1.^2))),-sqrt((2-t2.^2)./acos(t2./(1+t2.^2)))];
% 计算纵坐标数据
```

上面程序中的 x 和 y 就是凸轮轮廓的数据。此外还可以从函数 ezplot 绘制的曲线对象读取其中的数据，具体做法如下：

```
figure;ezplot('x^2+y^2*acos(x/(x^2+1))-2',[-2,2]);% 利用函数 ezplot 绘图
Ch=get(gca,'Children'); % 获取曲线对象对应的句柄
xd=get(Ch,'XData');   % 获取横轴数据
yd=get(Ch,'YData');   % 获取纵轴数据
```

这里 xd 和 yd 分别是横轴和纵轴的数据，这样就得到了凸轮的轮廓数据。可以选择足够多的数据，从而不必进行数据拟合操作。

25.2 阻尼运动

物体在运动过程中受各种阻力（如摩擦力、空气阻力等）的影响，出现能量逐渐衰减而导致运动减弱的现象，这种运动被称为阻尼运动。本节给出利用 MATLAB 模拟阻尼运动的两个例子。

例 25–1：给出如图 25.4 所示的一个弹簧振子，求解运动状态。

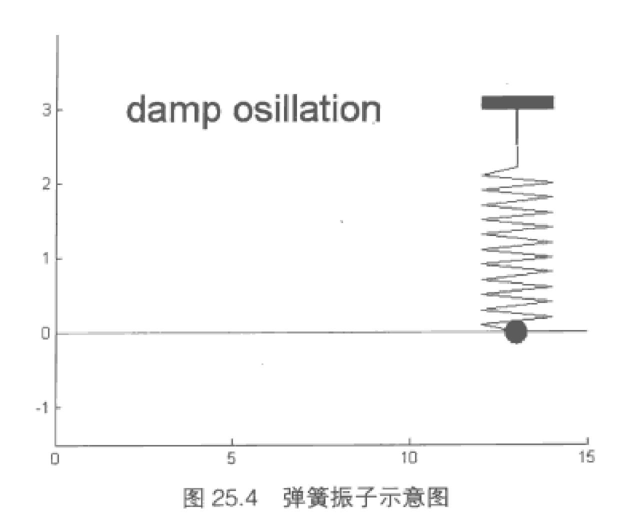


图 25.4 弹簧振子示意图

模拟弹簧阻尼振动过程包括以下内容：

- 更新小球和弹簧的位置。

◆ 更新水平线的位置和长度。

◆ 实时地画出弹簧高度与时间的关系曲线。

下面来模拟弹簧振子在阻尼力的作用下进行阻尼运动的过程，同时画出相应圆球质心随时间变化的曲线。相应的 MATLAB 程序如下：

```
rectangle('position',[12,8,2,0.3],'FaceColor',[0.1,0.3,0.4]); % 生成固定块
axis([0,15,-1,10]);hold on;      % 设置坐标轴范围
plot([13,13],[7,8],'r','linewidth',2); % 画与弹簧连接的线
y=2:.2:7;      % 得到弹簧对应的纵坐标数据
M=length(y); % 获取数据的长度
x=12+mod(1:M,2)*2;  % 生成弹簧的横坐标数据
x(1)=13;x(end-3:end)=13; % 计算出弹簧上下端点的横坐标值
D=plot(x,y);          % 画出弹簧
C=0:.1:2*pi; % 生成圆球的角度数据
r=0.3;      % 圆球的半径
t1=r*sin(C); % 计算出圆球对中心的纵坐标数据
F1=fill(13+r*cos(C),2+t1,'r');  % 画出圆球
set(gca,'ytick',[0:2:9]);  % 设置 Y 轴的刻度
set(gca,'yticklabels',num2str([-1:3]')); % 重新设置 Y 轴的刻度值
plot([0,15],[2,2],'black'); % 画出平衡位置
H1=plot([0,13],[2,2],'g'); % 球心的跟踪线
Q=plot(0,2.5,'color','r');  % 画出运动曲线
td=[]; % 记录时间的变量
yd=[]; % 记录 Y 轴位置的变量
T=0; % 设置初始的时间值
text(2,8,'damp osillation','fontsize',24); % 添加标注文字
set(gcf,'doublebuffer','on'); % 设置渲染效果
while T<12; % 利用循环处理阻尼运动过程的模拟
   pause(0.2); % 暂停一下以显示动画效果
   Dy=1-0.5*exp(-T/4)*cos(pi*T); % 计算 T 时刻弹簧对平衡位置的位移
   Y=-(y-2)*Dy+7; % 计算弹簧的纵坐标数值
   Yf=Y(end)+t1; % 计算圆球的纵坐标数值
   td=[td,T];yd=[yd,Y(end)]; % 更新运动曲线的数据
   set(D,'ydata',Y); % 更新弹簧的位置数据
   set(F1,'ydata',Yf,'facecolor',rand(1,3)); % 更新圆球的位置数据
   set(H1,'xdata',[T,13],'ydata',[Y(end),Y(end)]); % 更新跟踪线的数据
   set(Q,'xdata',td,'ydata',yd); % 更新圆球的数据
   T=T+0.1; % 更新时间
end
Kd=find(diff(sign(diff(yd)))==-2)+1; % 计算极大值的位置
X=td(Kd); % 得到极大值处的横坐标数值
Y=yd(Kd); % 得到极大值处的纵坐标数值
X=[0,X,td(end)]; % 得到上侧包络线的横坐标数据
Y=[yd(1),Y,yd(end)]; % 得到上侧包络线的纵坐标数据
plot(X,Y,':'); % 画出上侧包络线
Kx=find(diff(sign(diff(yd)))==2)+1;  % 计算极小值的位置
X=td(Kx); % 得到极小值处的横坐标数值
Y=yd(Kx); % 得到极小值处的纵坐标数值
X=[0,X,td(end)]; % 得到下侧包络线的横坐标数据
Y=[-(yd(1)-4),Y,-(yd(end)-4)]; % 得到下侧包络线的纵坐标数据
plot(X,Y,':'); % 画出下侧包络线
```

在循环计算之前把弹簧、固定杆、小球、水平线以及轨迹的初始图形对象绘制出来，其中随时

间变化的图形对象需要设置相应的句柄，以便在不同时刻利用函数 set 更新它们的位置和形状等信息。随后可以利用循环结构计算不同时刻系统各部分的位置，并且计算出小球高度随时间变化的曲线。上述程序模拟了弹簧振子进行阻尼运动的过程，其过程中的一个截图如同 25.5 所示。其中轨迹线和圆球质心相连的水平线随着时间的增加而缩短，这样生成的轨迹线就如同这条水平线“画”出来的效果一样。当程序执行结束后可以得到如图 25.6 所示的图形，其中给出了阻尼运动轨迹的包络线，可见振幅随着时间的增加而缩小。

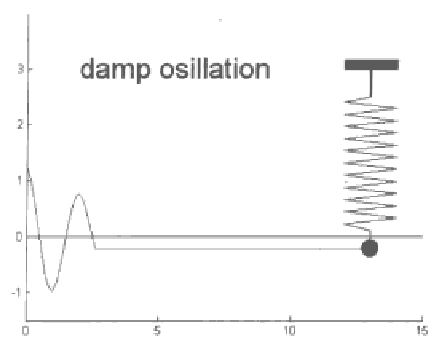


图 25.5 弹簧振子进行阻尼运动的模拟过程截图

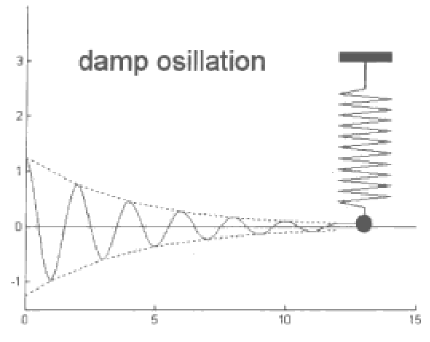


图 25.6 弹簧振子模拟的曲线及包络线

例 25–2： 小球受阻尼力作用下圆周运动，同时小球的大小随着时间以 e 指数的规律减小。模拟这个运动过程需要含以下三方面的内容：

◆ 计算小球在不同时刻的位置。
◆ 记录小球运动形成的轨迹。
◆ 实时更新小球的大小。

通过上述三方面的模拟可以得到这个过程的动画显示，相应的 MATLAB 程序如下：

```
n=1; % 阻尼系数
w=6; % 圆频率
ww=sqrt(w^2-n^2); % 等效的圆频率
B1=80; % 系数
B2=-20; % 系数
axis([-105,105,-105,105]);hold on; % 设置坐标轴范围
set(gcf,'doublebuffer','on'); % 设置图形窗口的渲染效果
G=plot(100,0,'r'); % 轨迹曲线
x=100; % 初始位置的横坐标
y=0; % 初始位置的纵坐标
r=2; % 小球半径
t1=r*sin(0:.1:2*pi); % 计算小球的边界数据
t2=r*cos(0:.1:2*pi); % 计算小球的边界数据
X=x+r*t1; % 得到小球的横坐标数据
Y=y+r*t2; % 得到小球的纵坐标数据
Q=fill(X,Y,rand(1,3)); % 画出小球
t=0; % 时间的初始数值
plot([-100,100],[0,0]); % 画出格线
plot([0,0],[-100,100]); % 画出格线
k=1; % 控制循环的参数
xlabel('Please press "space" key and stop this program!',...
   'fontsize',14,'color','r'); % 添加提示字符
```

```
title(['《damp turbination》  ','B1=',num2str(B1),...
      ';  B2=',num2str(B2)],'fontsize',14); % 加注图题
while k; % 循环处理
   s=get(gcf,'currentkey'); % 获取当前键入的按键名称
   if strcmp(s,'space'); % 判断按键是否为空格键
       clc;k=0; % 把参数 k 设置为 0，从而可以结束循环
   end
   pause(0.3); % 暂停一下显示动画效果
   z=exp(-n*t)*(B1*exp(i*ww*t)+B2*exp(-i*ww*t)); % 计算当前时刻的位置
   xp=real(z); % 分离出实部数值
   yp=imag(z); % 分离出虚部数值
   r=0.5+1.5*exp(-n*t); % 计算当前小球的半径
   X=xp+t1*r; % 计算小球的横坐标数据
   Y=yp+t2*r; % 计算小球的纵坐标数据
   set(Q,'xdata',X,'ydata',Y); % 更新小球的数据
   x=[x,xp]; % 存储轨迹的横坐标数据
   y=[y,yp]; % 存储轨迹的纵坐标数据
   set(G,'xdata',x,'ydata',y); % 更新轨迹曲线的数据
   t=t+0.02; % 更新时间数值
end
```

初始时刻把小球放置在水平轴离圆心一定位置处，假设小球开始时向上运动。此处小球的绘制是通过 fill 函数填充圆形区域实现的。随后计算小球在不同时刻的位置，相应地更新小球位置及其轨迹。此外在小球运动过程中设置小球的颜色随机变化。程序的终止是通过按空格键来控制的。运行程序得到 25.7 所示的图形。可见小球最终停留在原点附近。

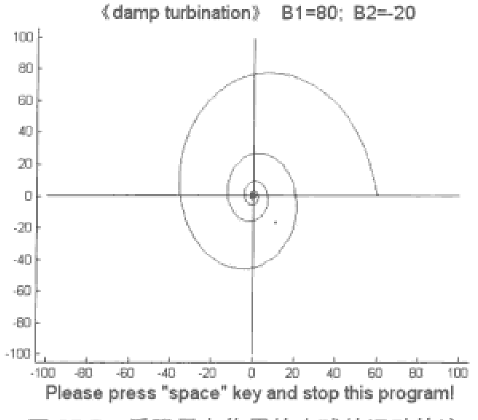


图 25.7　受阻尼力作用的小球的运动轨迹

25.3　连杆机构的运动模拟

很多机械结构中包含连杆结构，它们经常进行圆周运动或圆弧运动，同时可以带动其他结构相应地运动。在模拟连杆结构运动时可以利用几何知识来确定特征点的位置，从而确定连杆的位置。本节介绍连杆结构的机械运动模拟。

25.3.1　双摆运动的模拟

本节给出双摆运动的模拟。双摆运动可以由下面的微分方程组来描述。

$$\begin{cases} x_1' = x_3 \\ x_2' = x_4 \\ x_3' = A(1) \\ x_4' = A(2) \end{cases} \tag{25-3}$$

其中 A 表示一个列向量，由下式计算。

$$A = \begin{bmatrix} (m_1+m_2)L_1 & m_2L_2\cos(x_1-x_2) \\ m_1L_1\cos(x_1-x_2) & m_1L_2 \end{bmatrix}^{-1} \begin{bmatrix} m_2L_2x_4^2\sin(x_2-x_1)-(m_1+m_2)g\sin x_1 \\ m_2L_1x_3^2\sin(x_1-x_2)-m_2g\sin x_2 \end{bmatrix} \tag{25-4}$$

其中 m_1 和 m_2 分别表示上摆和下摆的质量，L_1 和 L_2 分别表示上摆和下摆的长度，g 表示重力加速度，x_1 和 x_2 分别表示上摆和下摆角位移，x_3 与 x_4 分别表示上摆和下摆的角速度。对于上面的微分方程组可以使用 ode45 函数来求解相应的 MATLAB 程序。

```
m1=1; % 上摆质量
m2=1; % 下摆质量
L1=1; % 上摆长度
L2=1; % 下摆长度
g=9.8;% 重力加速度
Da=inline(['[x(3);x(4);',...
    'inv([(m1+m2)*L1,m2*L2*cos(x(1)-x(2));',...
    'm1*L1*cos(x(1)-x(2)),m1*L2])*'...
    '[m2*L2*x(4)^2*sin(x(2)-x(1))-(m1+m2)*g*sin(x(1));',...
    'm2*L1*x(3)^2*sin(x(1)-x(2))-m2*g*sin(x(2))]]'],'t','x',...
    'flag','m1','m2','L1','L2','g'); % 定义微分方程组
set(gcf,'DoubleBuffer','on'); % 设置图形窗口的渲染效果
[t,x]=ode45(Da,[0,20],[0.8,0.8,0,0],[],m1,m2,L1,L2,g);
axis([-(L1+L2),(L1+L2),-(L1+L2)*1.8,1]); % 设置坐标轴范围
axis square;hold on; % 设置坐标轴属性
gh1=plot([0,L1*exp(i*(x(1)-pi/2))],'r-'); % 绘制上面的单摆
set(gh1,'linewidth',2,'markersize',6,'marker','o'); % 设置线宽和标识符号
gh2=plot([L1*exp(i*(x(1)-pi/2)),L1*exp(i*(x(1)-pi/2))+L2*exp(i*(x(2)-pi/2))],'
b-'); % 绘制下面的单摆
set(gh2,'linewidth',2,'markersize',6,'marker','o'); % 设置线宽和标识符号
for k=2:size(x,1); % 制作关于双摆的动画
 C1=[0,L1*exp(i*(x(k,1)-pi/2))]; % 计算上摆的两端点坐标
 C2=[L1*exp(i*(x(k,1)-pi/2)),L1*exp(i*(x(k,1)-pi/2))+L2*exp(i*(x(k,2)-pi/2))];
% 计算下摆的两端点坐标
    set(gh1,'xdata',real(C1),'ydata',imag(C1)); % 更新上摆的位置坐标
    set(gh2,'xdata',real(C2),'ydata',imag(C2)); % 更新下摆的位置坐标
    title(['t= ',num2str(t(k))],'fontsize',12); % 显示时间
    pause(0.1);
end
figure;
subplot(231);plot(t,x(:,1));title('t-\theta_1');xlabel('t');ylabel('\theta_1');
%上摆的角位移随时间变化
subplot(232);plot(t,x(:,2));title('t-\theta_2');xlabel('t');ylabel('\theta_2');
%下摆的角位移随时间变化
subplot(233);plot(t,x(:,3));title('t-\omega_1');xlabel('t');ylabel('\omega_1');
%上摆角速度随时间变化
subplot(234);plot(t,x(:,4));title('t-\omega_2');xlabel('t');ylabel('\omega_2');
%下摆角速度随时间变化
```

```
subplot(235);plot(x(:,1),x(:,3));title('\theta_1-\omega_1');xlabel('\theta_1')
;ylabel('\omega_1');
subplot(236);plot(x(:,2),x(:,4));title('\theta_2-\omega_2');xlabel('\theta_2')
;ylabel('\omega_2');
```

首先对双摆结构参数进行赋值，然后利用函数 ode45 求解公式（24-5）给出的微分方程，得到双摆端点在不同时刻的位置。然后根据端点的数据按时间顺序依次绘制不同时刻双摆连线而得到双摆运动过程的动画。上述程序可以模拟双摆运动的过程，图 25.8 给出了动画过程的一个截图。

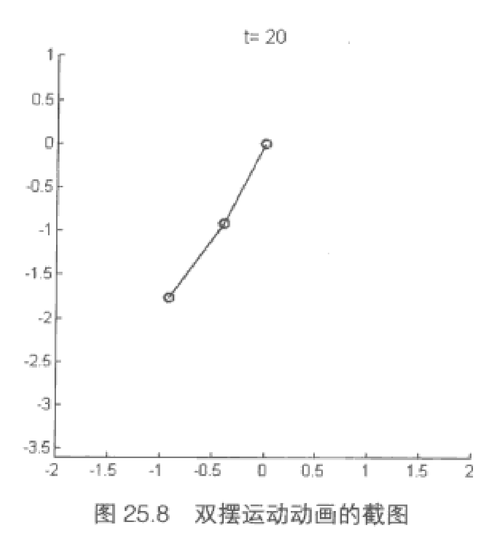


图 25.8　双摆运动动画的截图

如图 25.9 所示，是双摆运动角位移（上摆为 θ_1、下摆为 θ_2）、角速度（上摆为 ω_1、下摆为 ω_2）以及双摆的相图曲线（$\theta_1-\omega_1$ 曲线和 $\theta_2-\omega_2$ 曲线），读者从中可以了解双摆运动的状态。

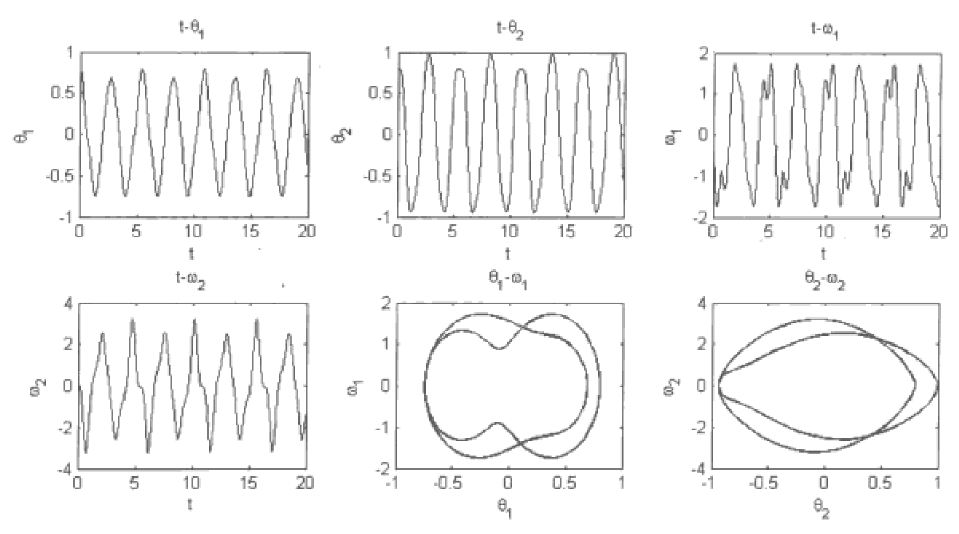


图 25.9　双摆运动参量随时间变化的曲线以及相图

25.3.2 四连杆结构的运动情况

本节模拟一个四连杆结构的运动情况，如图 25.10 所示。其中 O 点和 C 点是固定的，线段 OA 可以围绕着 O 点进行匀速圆周运动，这样就可以同时带动 B 点进行运动。

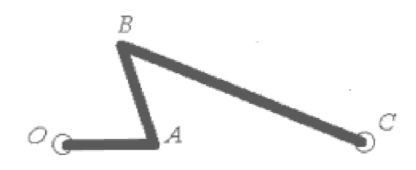


图 25.10　四连杆结构示意图

模拟这个运动过程的关键是实时地确定 A 点和 B 点的位置，其中 A 点随时间变化的位置可以确定。这样就可以计算出 A 点与 C 点之间的距离。在三角形 ABC 中，AB 边和 BC 边的长度是不随时间变化的，因此可以利用余弦定理计算出在任意时刻 AB 边与 AC 边的夹角，从而确定 B 点的坐标。下面考虑这个过程的模拟，相应的 MATLAB 程序如下：

```
t1=sin(0:.1:2*pi); % 生成固定点的横轴数据
t2=cos(0:.1:2*pi); % 生成固定点的纵轴数据
r=0.1; % 固定点的半径
fill(r*t1,r*t2,'r');set(gcf,'doublebuffer','on'); % 绘制左侧固定点
hold on; fill(2+r*t1,r*t2,'r'); % 绘制右侧固定点
axis([-1,6,-1,4]);axis equal; % 设置坐标轴范围及属性
t=0; % 记录时间的参数
A=t; % 旋转的角度
z=0.5*exp(i*A); % 圆周运动的端点坐标
H1=plot([0,0.5*cos(A)],[0,0.5*sin(A)],'linewidth',4); % 画出圆周运动的连杆 OA
AC=abs(2-z); % 计算当前线段 AC 的长度
AB=0.7; % 设置 AB 的长度，其为固定值
BC=1.8; % 设置 BC 的长度，其为固定值
JA=acos((AC^2+AB^2-BC^2)/(2*AC*AB)); % 计算 AB 和 BC 夹角大小
B=z+AB*exp((angle(2-z)+JA)*i); % 获得 B 点当前的坐标
k=1; % 控制循环是否执行的参数
H2=plot(real([B,z]),imag([B,z]),'g','linewidth',4); % 绘制 AB 连杆
H3=plot(real([B,2]),imag([B,2]),'linewidth',4,'color',[0.4,0.2,0.5]);  % 绘制 BC
连杆
Ga=plot(z); % 绘制 A 点当前的轨迹曲线
Gb=plot(B); % 绘制 B 点当前的轨迹曲线
set([Ga,Gb],'color','r'); % 设置轨迹曲线的颜色
zga=z; % 定义记录 A 点轨迹点的变量
zgb=B; % 定义记录 B 点轨迹点的变量
xlabel('Please press "space" key and stop this
program!','fontsize',14,'color','r'); % 加注标注
T=title(['time=  ',num2str(t)]); % 实时显示时间
text(0,2.5,'四连杆机构'); % 加注文字
while k; % 模拟四连杆的运动过程
   s=get(gcf,'currentkey'); % 检测键入按键的名称
   if strcmp(s,'space'); % 判断键入按键是否为空格键
      clc;k=0; % 设置参数使循环结束
   end
   pause(0.01); % 暂停一下，显示动画效果
   t=t+0.02; % 更新时间
   z=0.5*exp(i*t); % 计算 A 点实时位置
   AC=abs(2-z); % 计算当前线段 AC 的长度
   JA=acos((AC^2+AB^2-BC^2)/(2*AC*AB)); % 计算当前线段 AB 和 AC 的夹角
   B=z+AB*exp((angle(2-z)+JA)*i); % 计算当前 B 点的坐标
   set(H1,'xdata',[0,0.5*cos(t)],'ydata',[0,0.5*sin(t)]); % 更新线段 OA 的位置
   set(H2,'xdata',real([B,z]),'ydata',imag([B,z])); % 更新线段 AB 的位置
   set(H3,'xdata',real([B,2]),'ydata',imag([B,2])); % 更新线段 BC 的位置
```

```
    if t<8;
        zga=[zga,z]; % 更新A点轨迹的数据集合
        zgb=[zgb,B]; % 更新B点轨迹的数据集合
        set(Ga,'xdata',real(zga),'ydata',imag(zga)); % 更新A点轨迹
        set(Gb,'xdata',real(zgb),'ydata',imag(zgb)); % 更新B点轨迹
    end
    set(T,'string',['time=  ',num2str(t)]); % 更新时间
end
```

首先初始化系统的参数，给出 O，A，B 和 C 四点的位置。利用函数 plot 绘制出三段连杆图形。接下来可以模拟 A 点进行圆周运动，利用余弦定理计算出不同时刻 B 点的位置，并相应地更新三条连杆的位置。此外，B 点的轨迹被实时地记录和显示。上述程序执行后得到如图 25.11 所示的图形，可以按下空格键来停止这个程序的执行。在图 25.11 中还显示了两个移动点的轨迹。

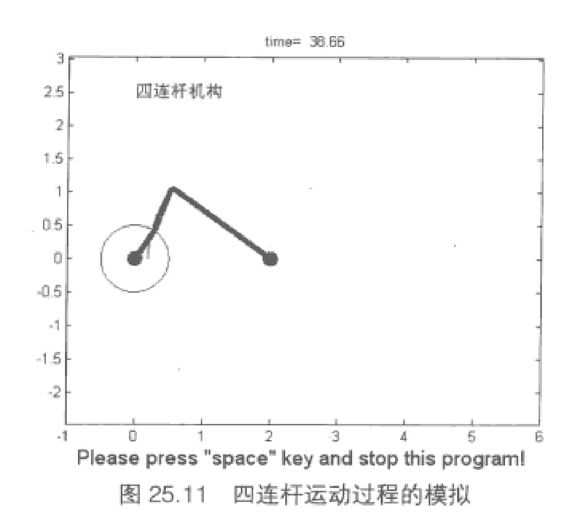


图 25.11　四连杆运动过程的模拟

25.3.3　带有套环的机械结构的运动过程

下面研究 1 个带有套环的机械结构的运动过程。整个系统的几何关系如图 25.12 所示，其中点 P_1，P_2 和 P_3 是被固定的，点 P_2 和 P_3 之间有一细杆相连且处于水平方向上。线段 P_1C 表示的连杆可以绕点 P_1 转动。杆 BD 两端是自由的，但是 B 点只能在线段 P_2P_3 上运动。线段 P_1C 与水平方向上的夹角等于 α，在整个运动过程中，线段 P_1C 与线段 BD 保持相互垂直的关系。

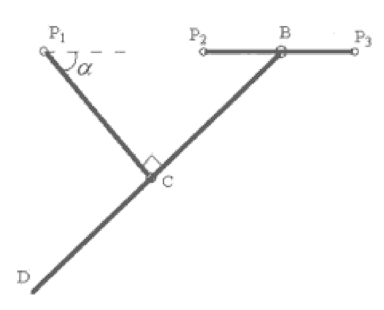


图 25.12　套环结构的机械运动

下面模拟连杆 P_1C 在一定的角度范围内运动时连杆 P_1C 和 BD 的运动情况，设初始时角度 $\alpha=\pi/3$，相应的 MATLAB 程序如下：

```
r=0.1; % 圆圈的半径
set(gcf,'doublebuffer','on'); % 设置图形渲染效果
fill(r*sin(0:.1:2*pi),r*cos(0:.1:2*pi),'b'); % 画出固定点 P1
hold on; axis equal; % 设置坐标轴属性
axis([-1,6,-6,1]);    % 设置坐标轴范围
fill(r*sin(0:.1:2*pi)+2,r*cos(0:.1:2*pi),'b'); % 画出固定点 P2
fill(r*sin(0:.1:2*pi)+5,r*cos(0:.1:2*pi),'b'); % 画出固定点 P3
plot([2,5],[0,0],'k','linewidth',3); % 画出水平横杆 P2-P3
A=-pi/6; %设置初始时倾斜角
z=2.85*exp(i*A); % 计算出初始时 C 点的坐标
zQ=0.15*exp(i*[0:.1:pi*2]); % 圆环的对中心的坐标数据
HL=plot([0,real(z)],[0,imag(z)],'r','linewidth',2); % 画出线段 P1-C
HQ=plot(3*exp(i*A)+zQ,'g'); % 画出 C 处的圆环
L=3/cos(A); % 计算出当前线段 P1-B 的长度
La=L+0.1*exp(i*(-pi/2+A)); % 计算当前 B 点的坐标
Lb=L+5*exp(i*(-pi/2+A)); % 计算当前 D 点的坐标
HLD=plot([La,Lb],'b','linewidth',6); % 画出当前线段 BD
HQQ=plot(L+zQ*2/3,'k'); % 画出 B 点处的圆圈
xlabel('Please press "space" key and stop this program!',...
   'fontsize',14,'color','r'); % 添加标注文字
HT=text(3,-5,'\it 套环运动','fontsize',18,'fontname','宋体'); % 添加标注文字
k=1; % 控制循环执行的参数
T=0; % 时间数值
dt=0.1; % 时间增量
Tr=Lb; % 轨迹数据
Htr=plot(real(Tr),imag(Tr),'m.','markersize',2); % 画出轨迹
while k; % 模拟运动过程
   pause(0.2); % 暂停一下，显示动画效果
   s=get(gcf,'currentkey'); % 获取键入按键的名称
   if strcmp(s,'space'); % 检查按下的按键是否为空格键
      clc;k=0; % 清屏，同时停止程序
   end
   T=T+dt*rand; % 更新时间，这里使用一个随机的增量
   Aa=A+sin(T)/3; % 计算出当前的角度
   L=3/cos(Aa); % 计算出当前 B 点的位置
   La=L+0.1*exp(i*(-pi/2+Aa)); % 给出线段 BD 端点 B 的坐标
   Lb=L+5*exp(i*(-pi/2+Aa)); % 计算出线段 BD 端点 D 的坐标
   set(HLD,'xdata',real([La,Lb]),'ydata',imag([La,Lb])); % 更新线段 BD 的位置
   set(HQQ,'xdata',real(L+zQ*2/3),'ydata',imag(L+zQ*2/3)); % 更新 B 点处圆环的位置
   z=2.85*exp(i*Aa); % 计算 C 点当前位置
   set(HL,'xdata',real([0,z]),'ydata',imag([0,z])); % 更新线段 P1-C 的位置
   set(HQ,'xdata',real(3*exp(i*Aa)+zQ),'ydata',...
      imag(3*exp(i*Aa)+zQ)); % 更新 C 点处的圆环位置
   set(HT,'color',rand(1,3)); % 更新标注文字的颜色
   if T<200; % 在时间小于 200 时记录轨迹位置
      Tr=[Tr,Lb]; % 把当前 D 点坐标数值记录下来
   end
   set(Htr,'XData',real(Tr),'YData',imag(Tr)); % 更新轨迹的坐标数据
end
```

首先给出该系统中小圆环、3 条连杆绘制及其长度的赋值，其中运动的杆对应的图形对象需要有句柄输出，便于以后更新位置。然后利用循环结果计算不同时刻 C 点和 D 点的位置，从而可以确定运动杆的位置。同时小环的位置也需要更新。在计算过程中，把 D 点不同时刻的位置记录下

来绘制成轨迹。上述程序模拟过程的一个截图如图 25.13 所示。同样可以键入空格键来停止这个程序的执行。

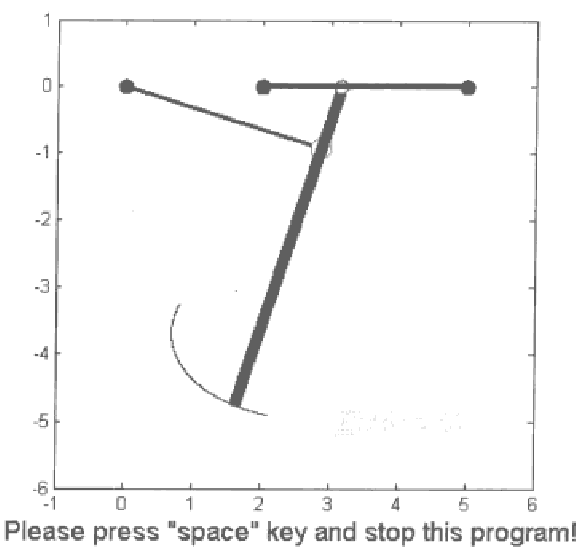


图 25.13 带有套环结构的机械运动模拟

25.3.4 小球在水平面上受 3 根弹簧作用下的运动情况

下面来模拟 1 个小球在水平面上受 3 根弹簧作用下的运动情况，如图 25.14 所示。假设小球所在平面足够光滑，因此可以认为小球不受摩擦力的作用。初始时把小球从平衡位置处拉开，并放开小球，它将在 3 根弹簧的作用下进行往复运动。

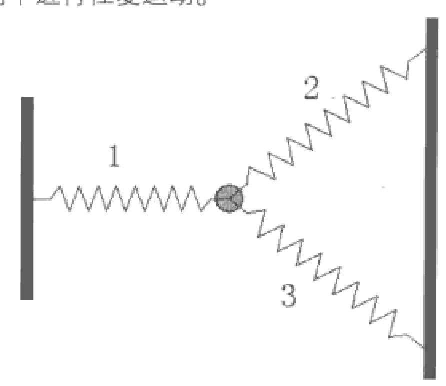


图 25.14 小球在 3 根弹簧作用下的模型

相应的模拟程序描述如下。

```
u=0.0; % 摩擦系数
M=0.2; % 振子质量
g=9.8; % 重力加速度
axis([0,10,0,8]); % 设置坐标轴范围
hold on;box on; % 设置坐标轴属性
F1=fill([1,1.2,1.2,1],[2,2,6,6],[0.8,0.2,0.8]); % 画出左侧固定杆的图案
F2=fill([8.8,9,9,8.8],[0.4,0.4,7.6,7.6],[0.8,0.2,0.8]); % 画出右侧固定杆的图案
p1=1.2+4i; % 给出弹簧 1 在左侧固定条上的端点
```

```
p2=8.8+7i; % 给出弹簧 2 在右侧固定条上的端点
p3=8.8+i; % 给出弹簧 3 在右侧固定条上的端点
k1=10; % 第 1 个弹簧的系数
k2=10; % 第 2 个弹簧的系数
k3=10; % 第 3 个弹簧的系数
pz=5+4i; % 小球的初始位置
Nk=[mod(1:16,2)-0.5]*0.5; % 计算弹簧 1 的位置
Nk=[0,0,0,Nk,0,0,0];
pd=pz+exp(i*[0:0.02:pi*2])*0.26; % 计算小球边缘的坐标
Fq=fill(real(pd),imag(pd),[0.7,0.8,0.9]); % 画出小球
N1=exp(i*[angle(p1-pz)+pi/2]); % 计算与弹簧 1 轴线垂直的方向
Pk1=linspace(pz,p1,22); % 生成 pz 和 p1 之间等间距的序列
Pk1=Pk1+Nk*N1; % 计算弹簧 1 对应的数据点
hp1=plot(Pk1); % 画出第 1 个弹簧
N2=exp(i*[angle(p2-pz)+pi/2]); % 计算与弹簧 2 轴线垂直的方向
Pk2=linspace(pz,p2,22); % 生成 pz 和 p2 之间等间距的序列
Pk2=Pk2+Nk*N2; % 计算弹簧 2 对应的数据点
hp2=plot(Pk2); % 画出第 2 个弹簧
N3=exp(i*[angle(p3-pz)+pi/2]); % 计算与弹簧 3 轴线垂直的方向
Pk3=linspace(pz,p3,22); % 生成 pz 和 p3 之间等间距的序列
Pk3=Pk3+Nk*N3; % 计算弹簧 2 对应的数据点
hp3=plot(Pk3); % 画出第 3 个弹簧
L1=abs(p1-pz); % 计算出弹簧的原来长度
L2=abs(p2-pz); % 计算出弹簧的原来长度
L3=abs(p3-pz); % 计算出弹簧的原来长度
v=0; % 初始时的速度
set(gcf,'DoubleBuffer','on'); % 设置图形的渲染效果
dt=0.01; % 时间增量
Tz=title('t=0'); % 显示当前时间
t=0; % 初始时间数值
% Rp=[rand-0.5+i*(rand-0.5)]*4; % 初始相对位置（随机）
Rp=2+2i;                        % 初始相对位置（自行设定）
pz=pz+Rp; % 记算当前位置
pzz=pz; % pzz 是记录小球轨迹的变量
F=1;
hG=plot(pz,'r.','markersize',2);  % 绘制轨迹
while abs(v)>1e-5 | abs(F)>1e-4; % 循环计算小球运动过程
    F1=exp(i*[angle(p1-pz)])*(abs(p1-pz)-L1); % 计算弹簧 1 产生的弹力
    F2=exp(i*[angle(p2-pz)])*(abs(p2-pz)-L2); % 计算弹簧 2 产生的弹力
    F3=exp(i*[angle(p3-pz)])*(abs(p3-pz)-L3); % 计算弹簧 3 产生的弹力
    F=F1+F2+F3; % 计算 3 个弹簧的合力
    a=F/M; % 计算小球当前的加速度
    v=v+a*dt; % 计算小球当前的速度
    pz=pz+v*dt; % 计算小球当前的位移
    N1=exp(i*[angle(p1-pz)+pi/2]); % 计算与弹簧 1 轴线垂直的方向
    Pk1=linspace(pz,p1,22); % 生成 pz 和 p1 之间等间距的序列
    Pk1=Pk1+Nk*N1; % 计算第 1 个弹簧的坐标数值
    set(hp1,'XData',real(Pk1),'YData',imag(Pk1)); % 更新第 1 个弹簧的位置
    N2=exp(i*[angle(p2-pz)+pi/2]); % 计算与弹簧 2 轴线垂直的方向
    Pk2=linspace(pz,p2,22); % 生成 pz 和 p2 之间等间距的序列
    Pk2=Pk2+Nk*N2; % 计算第 2 个弹簧的坐标数值
    set(hp2,'XData',real(Pk2),'YData',imag(Pk2)); % 更新第 2 个弹簧的位置
    N3=exp(i*[angle(p3-pz)+pi/2]); % 计算与弹簧 3 轴线垂直的方向
    Pk3=linspace(pz,p3,22); % 生成 pz 和 p3 之间等间距的序列
    Pk3=Pk3+Nk*N3; % 计算第 3 个弹簧的坐标数值
    set(hp3,'XData',real(Pk3),'YData',imag(Pk3)); % 更新第 3 个弹簧的位置
```

```
    pd=pz+exp(i*[0:0.02:pi*2])*0.26; % 计算小球当前位置
    set(Fq,'XData',real(pd),'YData',imag(pd)); % 更新小球的位置
    t=t+dt; % 更新时间数值
    if t<800; % 在 t 小于 800 时记录轨迹点
        pzz=[pzz,pz]; % 记录当前小球位置数据到变量 pzz 中
        set(hG,'Xdata',real(pzz),'YData',imag(pzz)); % 更新轨迹图
    end
    set(Tz,'string',['t=',num2str(t)]); % 更新当前时间
    pause(0.02); % 暂停一下显示动画效果
end
```

首先初始化系统的参数，如位置和长度等信息。在该系统中运动的组件有 3 根弹簧和 1 个小球。而在计算中只需要确定小球的质心就可以确定运动组件的位置。小球的运动情况可以通过受力分析来计算，此处小球仅受到弹簧弹力作用，计算出合力就可以得到小球的加速度，继而得到任意时刻小球的速度，通过速度可以计算出小球的位置。另外，程序中把小球轨迹记录下来并显示。相应的运动过程如图 25.15 所示。在图 25.15 中间部分的网格状复杂图形是小球质心的运动轨迹，可见这是一个复杂的轨迹图。

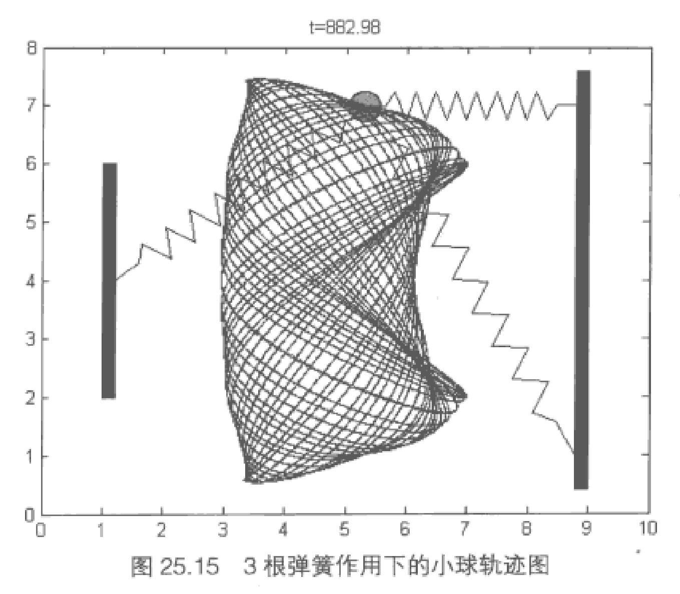


图 25.15　3 根弹簧作用下的小球轨迹图

此外，光盘中\Ch25 文件夹下的 spring11 给出了一个弹簧振子的动画模拟。一根弹簧被水平放置在地面上。弹簧的一端固定在墙上，另一端系着一个小球为自由端。这里忽略小球的摩擦力以及相关阻力，如此小球做简谐运动。

25.4　凸轮的转动

本节来研究凸轮匀速转动的一个例子。凸轮系统的几何结构如图 25.16 所示，其中凸轮被固定在 O_1 位置，它可以围绕 O_1 点旋转，而椭圆面被固定在 O_2 点。AB 为一个连杆，其长度等于 O_1 点和 O_2 点之间的距离。凸轮进行匀速转动的时候将会带动椭圆面进行转动。

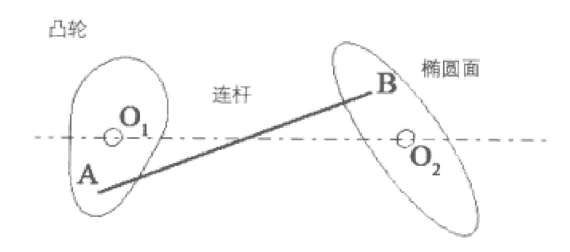


图 25.16 凸轮系统的构图

根据几何知识可知，A 点和 B 点将进行圆周运动。给出凸轮的旋转角度可以根据余弦定理推算出椭圆面的角度。这里凸轮的外轮廓根据前面等式（25-2）来计算。

下面给出这个过程模拟的 MATLAB 程序。

```
t1=linspace(-sqrt(2)+eps,sqrt(2)-eps,200); % 生成递增数据序列
t2=linspace(sqrt(2)-eps,-sqrt(2)+eps,200); % 生成递减数据序列
x1=[t1,t2]; % 生成凸轮轮廓的横坐标数据
y1=[sqrt((2-t1.^2)./acos(t1./(1+t1.^2))),-sqrt((2-t2.^2)./acos(t2./(1+t2.^2)))
]/1.5; % 凸轮轮廓的纵坐标
ph=plot(x1,y1,'r');% 画出凸轮的外轮廓
hold on;axis([-2,8,-4,4]);axis equal; % 设置坐标轴范围
a=linspace(0,pi*2,200); % 圆周角序列
plot(0.15*exp(i*a),'k');  % 画出左侧固定点附近的圆圈
x2=5+2*cos(a); % 椭圆边界的数据，横坐标
y2=0.6*sin(a);  % 椭圆边界的数据，纵坐标
eh=plot(x2,y2,'m'); % 画出椭圆的轮廓
plot(5+0.15*exp(i*a),'k'); % 画出右侧固定点附近的圆圈
R=1;    % 连杆右端点到右侧固定点的距离
pa=-1;   % 连杆左端点的坐标
pb=5-R;  % 连杆右端点的坐标
L=abs(pa-pb); % 计算连杆长度
Lh=plot([pa,pb],[0,0],'k','linewidth',2); % 画出连杆
plot([-2,8],[0,0],'k-.'); % 画出水平轴线
k=1;  % 控制循环的参数
A=pi; % 初始时凸轮的旋转角度
dA=0.1; % 角度增量
plot(exp(i*a),'k:');   % A 点的轨迹
plot(5+exp(i*a),'k:'); % B 点的轨迹
ti=title('angle = \pi'); % 添加图题
while k;
   A=A+dA;
   T1=[cos(A-pi),-sin(A-pi);sin(A-pi),cos(A-pi)]; % 坐标旋转矩阵
   pa=exp(i*A); % 更新连杆左端点数值
   x1t=T1(1,1)*x1+T1(1,2)*y1; % 计算当前凸轮轮廓的数据横坐标
   y1t=T1(2,1)*x1+T1(2,2)*y1; % 计算当前凸轮轮廓的数据纵坐标
   D=abs(pa-5); % 计算连杆左端点到右侧固定点的距离
   da=acos([R^2+D^2-L^2]/[2*R*D]); % 计算连线 O2-A 与 O2-B 的夹角
   A2=angle(pa-5); % 计算 A 点和 O2 点连线与水平方向的夹角
   A2=A2+sign(A2)*da; % 计算 B 点与 O2 点连线的夹角
   pb=R*exp(i*A2)+5;  % 计算出 B 点的坐标
   T2=[cos(A2-pi),-sin(A2-pi);sin(A2-pi),cos(A2-pi)]; % 坐标旋转矩阵
   x2t=T2(1,1)*[x2-5]+T2(1,2)*y2; % 计算当前凸轮轮廓的数据横坐标
   y2t=T2(2,1)*[x2-5]+T2(2,2)*y2; % 计算当前凸轮轮廓的数据纵坐标
```

```
    set(ph,'XData',x1t,'YData',y1t); % 更新凸轮轮廓的位置
    set(eh,'XData',[x2t+5],'YData',y2t); % 更新椭圆轮廓的位置
    set(Lh,'XData',real([pa,pb]),'YData',imag([pa,pb])); % 更新连杆的位置
    set(ti,'String',['angle = ',num2str(mod(A,pi*2))]); % 显示当前角度
    pause(0.2); % 暂停一下，显示动画效果
end
```

首先初始化凸轮系统的组件的形状参数，并利用 plot 函数画出凸轮轮廓图和椭圆面。然后利用循环结构计算不同时刻该系统的位置，首先根据时间确定凸轮位置，继而可以得到 A 点的坐标，然后利用余弦定理可以计算出 B 点的坐标，再根据 B 点的坐标确定椭圆面的位置并更新图形。通过理论计算可以知道 A 点和 B 点的轨迹是圆，这样在计算之前把两个圆用虚线画出来。上述程序给出了这个转动过程的模拟，其中一个截面如图 25.17 所示，图中的两个虚线圆圈表示连杆两个端点运动的轨迹。

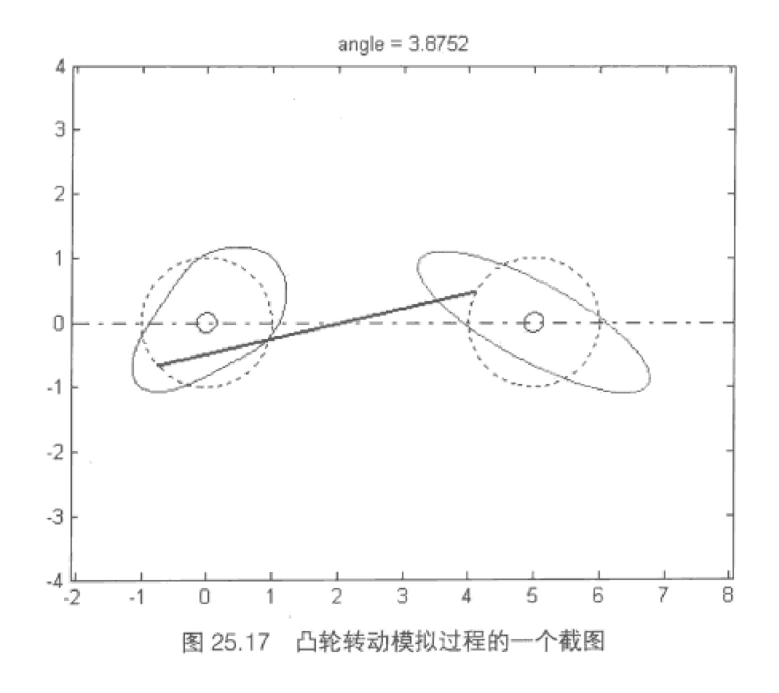


图 25.17　凸轮转动模拟过程的一个截图

25.5 小结

机械运动的模拟主要通过几何学知识建立系统各个组件之间在不同时刻的位置关系。此外，必要的力学知识可用来建立机械运动的方程。本章结合 MATLAB 的数值计算和数据的可视化功能模拟了几种形式的机械运动。通过这些例子读者可以模仿其他形式的机械运动。本章首先给出了一个凸轮围绕其轴线旋转的例子，随后给出了阻尼运动模拟，以及双摆、四连杆、套环、三弹簧振子等运动形式的模拟实例，最后给出了凸轮带动椭圆面旋转的例子。

◆　◆　◆

第 26 章　经济和金融问题的求解

本章包括

- **金融工具箱介绍**　介绍金融工具箱中部分函数的用途。
- **时间序列预测模型**　介绍几种生成时间序列的预测模型。
- **经济学模型**　介绍 4 种经济学模型的数学定义及相关的计算。
- **规划问题求解**　介绍生产和经营管理中遇到的线性规划问题的求解。

MATLAB 提供了金融工具箱用于分析金融数据以及开发金融算法。金融工具箱构建于 MATLAB 以及统计和优化工具箱功能之上，该工具箱提供广泛的金融特定函数支持。金融工具箱使读者能够优化投资组合、确定风险、分析利率水平、对股权衍生物定价、处理和转化商业日期，同时包括丰富的工具进行日期处理、货币的格式化、金融数据的图形显示等。本章结合这个工具箱介绍 MATLAB 在经济学中的应用。

26.1　金融工具箱介绍

本节来介绍 MATLAB 金融工具箱中的一些基本函数。MATLAB 金融工具箱的相关文件存放在安装路径的子目录\MATLABR2008a\toolbox\finance 下。

MATLAB 提供的投资组合分析方面的函数如表 26.1 所示。计算利率期限结构方面的函数如表 26.2 所示。计算期权评估以及敏感度方面的函数如表 26.3 所示。计算金融效率相关的函数如表 26.4 所示。

表 26.1　投资组合分析方面的函数

函数名	功能说明	函数名	功能说明
portsim	多资产回报序列的蒙特卡洛模拟	portopt	任意约束条件的边界条件
portalloc	效率前缘投资组合的资本分配	portvrisk	计算投资组合风险值

表 26.2　利率期限结构方面的函数

函数名	功能说明	函数名	功能说明
disc2zero	把贴现曲线转化为零曲线	zbtprice	用 bootstrap 方法根据债券价格计算零曲线
fwd2zero	把正向曲线转化为零曲线	zbtyield	用 bootstrap 方法根据债券收益计算零曲线
prbyzero	从零曲线对债券定价	zero2disc	把零曲线转化为贴现曲线
pyld2zero	把平均收益曲线转化为零曲线	zero2fwd	把零曲线转化为正向曲线
termfit	用样条工具箱对期限结构拟合	zero2pyld	把零曲线转化为平均收益曲线

表 26.3　计算期权评估以及敏感度方面的函数

函数名	功能说明	函数名	功能说明
blkprice	用 Black scholes 方法进行期权定价	glsgamma	用 Black scholes 进行敏感度分析
blsprice	用 Black sholes 公式计算买入买权和卖出卖权值		

表 26.4　计算金融效率相关的函数

函数名	功能说明	函数名	功能说明
annurate	计算养老金周期利率	mirr	根据现金流计算修订内部回报率
annuterm	获取某价值所需的周期数	nomrr	计算实际回报率
effrr	计算回报的有效率	pvfix	计算固定周期支付的现值
irr	计算内部回报率		

这些函数的使用可以通过函数 help 参阅相关的帮助文档。

26.2　时间序列预测模型

时间序列（Time series）是指随着时间变化带有随机性且前后数据又有关联的一些数据流。在经济活动中产生的数据流在很多情况下都可以看做时间序列。因而时间序列在经济学的预测模型中占有重要的地位。以下将介绍布朗非线性指数法、Gomperta 曲线预测模型、Logistic 曲线预测模型等。

26.2.1　布朗（Brown）非线性指数法产生时间序列

本小节来介绍布朗（Brown）非线性指数法产生时间序列。如果时间序列呈现非线性趋势时，可以采用布朗非线性指数法进行时间序列的预测模拟。该方法的基本原理是在一次平滑和二次指数平滑的基础上再进行一次指数平滑而得到，即 3 次指数平滑。设时间序列为 x_1，x_2，…，x_N，相应的递推公式如下：

$$\begin{cases} S_1(t)=ax_t+(1-a)S_1(t-1) \\ S_2(t)=aS_1(t)+(1-a)S_2(t-1) \\ S_3(t)=aS_2(t)+(1-a)S_3(t-1) \end{cases} \tag{26-1}$$

一般情况下，取初始值为 $S_1(0)=S_2(0)=S_3(0)=x_1$。而布朗预测时间序列的公式为：

$$y(t+T)=k_1(t)+k_2(t)T+k_3(t)T^2 \tag{26-2}$$

其中 T 表示自 t 时刻起向前预测的时间长度，而系数 $k_1(t)$，$k_2(t)$ 和 $k_3(t)$ 通过下式计算：

$$\begin{cases} k_1(t)=3S_1(t)-3S_2(t)+S_3(t) \\ k_2(t)=\dfrac{a}{2(1-a)^2}[(6-5a)S_1(t)-(10-8a)S_2(t)+(4-3a)S_3(t)] \\ k_3(t)=\dfrac{a^2}{2(1-a)^2}[S_1(t)-2S_2(t)+S_3(t)] \end{cases} \quad (26\text{-}3)$$

假设时间序列$[x_1,x_2,...,x_{12}]=[50,52,47,51,49,48,51,40,48,51,51,59]$，这里取参数$a=0.3$来计算$y(12+T)$的数值。相关的 MATLAB 程序如下：

```
S1=50; % 设置初始值
S2=50; % 设置初始值
S3=50; % 设置初始值
x=[50,52,47,51,49,48,51,40,48,51,51,59]; % 计算时间序列
a=0.3; % 设置参数 a 的数值
for k=2:13;
    S1(k)=a*x(k-1)+(1-a)*S1(k-1); % 迭代计算
    S2(k)=a*S1(k)+(1-a)*S2(k-1); % 迭代计算
    S3(k)=a*S2(k)+(1-a)*S3(k-1); % 迭代计算
end
k1=3*S1(12)-3*S2(12)+S3(12); % 计算系数 k1
k2=a/[2*(1-a)^2]*[(6-5*a)*S1(12)-(10-8*a)*S2(12)+(4-3*a)*S3(12)]; % 计算系数 k2
k3=a/[2*(1-a)^2]*[S1(12)-2*S2(12)+S3(12)]; %  计算系数 k3
T=1:10; % 时间离散值
y = k1+k2*T+k3*T.^2; % 进行预测
plot(T,y,'*','MarkerSize',10); % 绘制数据
set(gca,'Position',[0.2,0.2,0.75,0.6]); % 设置坐标轴位置
xlabel('\itT','Fontsize',14,'Fontname','Times new roman'); % X 轴标注
ylabel('{\ity}(12+{\itT})','Fontsize',14,'Fontname','Times new roman'); % Y 轴标
注
```

计算所得时间序列数值和已知时间序列数值如表 26.5 所示。

表 26.5　时间序列数值表

序列	t										
	2	3	4	5	6	7	8	9	10	11	12
x_t	52	47	51	49	48	51	40	48	51	51	59
$S_1(t)$	50.0000	50.6000	49.5200	49.9640	49.6748	49.1724	49.7207	46.8045	47.1631	48.3142	49.1199
$S_2(t)$	50.0000	50.1800	49.9820	49.9766	49.8861	49.6719	49.6866	48.8219	48.3243	48.3213	48.5609
$S_3(t)$	50.0000	50.0540	50.0324	50.0157	49.9768	49.8853	49.8257	49.5246	49.1645	48.9115	48.8063

输出$y(12+T)$的图形如图 26.1 所示。

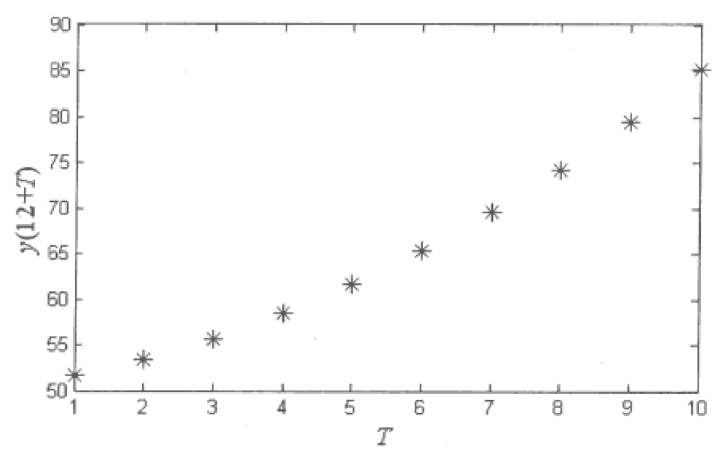


图 26.1 预测模型的图示结果

26.2.2 Gomperta 曲线预测模型

Gomperta 曲线是以统计学家龚伯兹的名字而命名的。该曲线描绘图形的特点是：开始增长速度很慢，随后逐渐加快，同时达到一定阶段变慢直到增长速度慢慢趋于 0。其曲线的走向很像一个顺时针倾斜的字母 S。如果已知的数据分布形式满足这个倾斜的 S 形，则可以考虑使用这个曲线模型进行预测研究。该曲线的数学方程为：

$$y = ka^{b^x} \tag{26-4}$$

其中 k ，a 和 b 为待定的系数。对方程（26-4）的两端取对数有：

$$\ln y = \ln k + b^x \ln a \tag{26-5}$$

在进行模型参数估计时，选用样本数据三等分并倒数求和的方法。采集一组样本数据 $(x_k, y_k)\ (k = 1,2,\dots,N)$，其中样本数据的个数为 3 的倍数，设 $r = N/3$，同时计算以下 3 个参数：

$$L_1 = \sum_{k=1}^{r} \ln y_k\ ,\quad L_2 = \sum_{k=r+1}^{2r} \ln y_k\ ,\quad L_3 = \sum_{k=2r+1}^{3r} \ln y_k \tag{26-6}$$

这里把样本数据按顺序分为三段，每段对应地算出一个参数 $L_k (k = 1,2,3)$。把公式（26-5）带入到公式（26-6）并整理后可得：

$$L_1 = r\ln k + b\frac{b^r - 1}{b-1}\ln a\ ,\quad L_2 = r\ln k + b^{r+1}\frac{b^r - 1}{b-1}\ln a\ ,\quad L_3 = r\ln k + b^{2r+1}\frac{b^r - 1}{b-1}\ln a \tag{26-7}$$

通过联立公式（26-7）中的 3 个等式，可以求解出待定参数 k ，a 和 b ，其结果为：

$$b = \sqrt[r]{\frac{L_3 - L_2}{L_2 - L_1}}\ ,\quad \ln a = (L_2 - L_1)\frac{b-1}{b(b^r - 1)^2}\ ,\quad \ln k = \frac{1}{r}\left[L_1 - \frac{b(b^r - 1)}{b-1}\ln a\right] \tag{26-8}$$

式（26-8）表明求解时需要按 $b \to a \to k$ 顺序进行。下面利用 Gomperta 曲线对表 26.6 中的数据进行预测。

表 26.6　某工厂的生产量统计表

时间（年）	2000	2001	2002	2003	2004	2005	2006	2007	2008
产量	41	51	71	166	248	329	360	381	399

上述样本数据个数正好是 3 的倍数。下面编写 Gomperta 曲线预测的程序，根据式（26-6）和（26-8）可以写出相应的 MATLAB 程序。

```
function yn=gomperta(y,xn);
% Gomperta 曲线预测的程序
r=fix(length(y)/3); % 历史数据 y 长度不是 3 的倍数时，取不大于其总数据的最大的 3 的整数倍数
L1=sum(log(y(1:r))); % 计算过程参数 L1
L2=sum(log(y(r+1:2*r))); % 计算过程参数 L2
L3=sum(log(y(2*r+1:3*r))); % 计算过程参数 L3
b=[(L3-L2)/(L2-L1)]^(1/r); % 计算待定系数 b
a=exp((L2-L1)*(b-1)/[b*(b^r-1)^2]); % 计算待定系数 a
k=exp([L1-b*(b^r-1)log(a)/(b-1)]/r); % 计算待定系数 k
yn=k*a.^(b.^xn); % 计算预测时间的数值
```

这是一个函数文件的形式，需要把上述内容保存为 gomperta.m 文件，之后就可以调用这个函数了，其调用格式为：

```
yn=gomperta(y,xn);
```

参数说明：yn 是预测的函数值。y 是输入的历史数据。xn 是待预测的位置数据。

利用下面一段程序可以用表 26.6 中的数据进行 Gomperta 曲线预测。

```
y=[41,51,71,166,248,329,360,381,399];% 输入历史数据
xn=10:12; % 待预测的位置
yn=gomperta(y,xn) % 进行 Gomperta 曲线预测
```

上述程序输出的结果为：

```
yn =
  427.1380  439.8634  448.6844
```

这 3 个数值依次对应着 2009 年、2010 年、2011 年的产量。



这里样本数据 x_k 默认为 1，2，…，9，它们对应着 1999 年、2000 年、…、2008 年，因此数值 10，11 和 12 对应的是 2009 年、2010 年和 2011 年。

26.2.3　logistic 曲线预测模型

logistic 曲线预测模型是由数学家 Veihulot 在研究人口增长规律时首先提出来的。该预测模型所定义的曲线图形与前面的 Gompertz 曲线相似。曲线的形状也是一个顺时针倾斜的 S 形。在很多情况下，这两种模型是可以互换使用的。logistic 曲线的数学方程为：

$$y=\frac{k}{1+me^{-at}} \tag{26-9}$$

其中 t 为时间序列的采样点，通常为自然数组成的数列。m 和 a 是待定参数。待定参数 k 表示函数 y 的数值增长趋势。当 $t\to\infty$ 有：

$$\lim_{t\to\infty} y = k \tag{26-10}$$

即 k 是函数 y 的饱和值。

下面介绍 3 个待定参数的计算方法。如果 k 已知或者可以根据实际情况计算出来，可以使用下面的方法计算参数 m 和 a 。式（26-9）可以改写为：

$$\frac{k}{y} - 1 = me^{-at} \tag{26-11}$$

对上式两边取对数有，得到

$$\ln\left(\frac{k}{y} - 1\right) = \ln m - at \tag{26-12}$$

如果记 $k/y-1=Y$ ，那么公式（26-12）是一个线性方程，通过线性回归分析可以计算待定参数 m 和 a 。

下面给出利用公式（26-12）编写 logistic 曲线预测模型的 MATLAB 程序。

```
function yn=logistic1(y,k,tn);
% logistic 曲线预测模型，计算参数 m 和 a
y=y(:); % 把历史数据转化为列向量
Y=k./y-1; % 计算合成变量 Y 的数值
n=numel(y); % 计算历史数据的个数
x=[[1:n]',ones(n,1)]; % 生成回归分析的输入参数 x
[B,Bint,R,Rint,Stats] = regress(Y,x); % 进行一元线性回归分析
if Stats(end)<0.01; % 判断回归分析结果是否是有效的
    m=exp(B(2)); % 计算参数 m
    a=-B(1); % 计算参数 a
    yn=k./(1+m*exp(-a*tn)); % 计算预期值
else
yn=[]; % 回归分析无效时，输出空的 yn
error('Error: The Logistic model is unsuccessful for the question.'); % 提示 logistic
曲线预测模型失效
end
```

上述程序是一个函数文件的形式，读者把上面这段程序保存为 logistic1.m 文件就可以对其调用了。该函数的调用格式为：

```
yn=logistic1(y,k,tn);
```

参数说明：yn 是计算所得的预测值。y 表示历史数据。k 是趋势值。tn 是待预测的时间位置。

下面以表 26.6 中的数据为例，调用函数 logistic1 对数据进行 logistic 曲线预测。这里选择参数 k 的数值为 410。通过下面的语句可以进行预测：

```
y=[41,51,71,166,248,329,360,381,399];% 输入历史数据
k=410; % 对参数 k 赋值
tn=10:12; % 待预测的位置
yn=logistic1(y,k,tn) % 进行 logistic 曲线预测模型
```

输出结果为：

```
yn =  390.6169  403.4756  407.8508
```

在 logistic 曲线预测模型中，预测的函数值 yn 要小于参数 k 的数值。

当参数 k 在进行预测前是未知时，可以采用样本数据三等分法。记样本数据数目为 $n=3r$，其中 r 是正整数。根据 logistic 曲线预测模型的数学表达式有：

$$\frac{1}{y_j}=\frac{1}{k}+\frac{m}{k}e^{-ja}，\ j=1,2,\ldots,n \tag{26-13}$$

然后对三段数据分别求和，设：

$$S_1=\sum_{j=1}^{r}\frac{1}{y_j}，\ S_2=\sum_{j=r+1}^{2r}\frac{1}{y_j}，\ S_3=\sum_{j=2r+1}^{3r}\frac{1}{y_j} \tag{26-14}$$

利用等比数列求和公式，联立公式（26-13）和（26-14），有：

$$S_1=\frac{r}{k}+\frac{m}{k}\frac{1-e^{-ra}}{e^{a}-1}，\ S_2=\frac{r}{k}+\frac{m}{k}\frac{e^{ra}-1}{\left(e^{a}-1\right)}，\ S_3=\frac{r}{k}+\frac{me^{ra}}{k}\frac{e^{ra}-1}{e^{a}-1} \tag{26-15}$$

令 $D_1=S_1-S_2$，$D_2=S_2-S_3$，进一步可以算出：

$$D_1=\frac{m}{k}\frac{\left(e^{-ra}-1\right)^2}{e^{a}-1}，\ D_2=\frac{m}{ke^{ra}}\frac{\left(e^{-ra}-1\right)^2}{e^{a}-1} \tag{26-16}$$

这样参数 a，k 和 m 的表达式如下：

$$a=\frac{1}{r}\ln\frac{S_1-S_2}{S_2-S_3}，\ k=\frac{\left(D_1-D_2\right)r}{\left(D_1-D_2\right)S_1-D_1^2}，\ m=\frac{D_1^2k}{D_2-D_1}\frac{e^{a}-1}{e^{-ra}-1} \tag{26-17}$$

根据上面的推导，建立 logistic 曲线预测模型的 MATLAB 程序，如下：

```
function yn=logistic2(y,tn);
% logistic 曲线预测模型，计算参数 k，m 和 a
n=numel(y); % 计算历史数据的个数
r=fix(n/3); % 历史数据 y 长度不是 3 的倍数时，取不大于其总数据的最大的 3 的整数倍数
S1=sum(1./y(1:r)); % 计算过程参数 S1
S2=sum(1./y(r+1:2*r)); % 计算过程参数 S2
S3=sum(1./y(2*r+1:3*r)); % 计算过程参数 S3
D1=S1-S2; % 计算过程参数 D1
D2=S2-S3; % 计算过程参数 D2
a=log([S1-S2]/[S2-S3])/r; % 计算待定参数 a
k=(D1-D2)*r/[(D1-D2)*S1-D1^2]; % 计算待定参数 k
m=D1^2*k/(D2-D1)*[exp(a)-1]/[exp(-r*a)-1]; % 计算待定参数 m
yn=k./(1+m*exp(-a*tn)); % 计算预期值
```

上述程序是一个函数文件的形式，读者把上面这段程序保存为 logistic2.m 文件就可以对其调用了。该函数的调用格式为：

```
yn = logistic2(y, tn);
```

参数说明：yn 是计算所得的预测值。y 表示历史数据。tn 是待预测的时间位置。

下面以表 26.6 中的数据为例，调用函数 logistic2 对数据进行 logistic 曲线预测。通过下面的语句可以进行预测：

```
y=[41,51,71,166,248,329,360,381,399];% 输入历史数据
tn=10:12; % 待预测的位置
yn=logistic2(y,tn) % 进行 logistic 曲线预测模型
```

输出结果为：

```
yn =  407.1473  411.0321  412.9498
```

26.3 经济学模型

经济学模型可以很好地刻画系统的参量变化过程，通过模型可以了解系统的稳定性情况，预测未来发展趋势等。本节来介绍几种经典的模型。

26.3.1 凯恩斯模型

凯恩斯模型是一种用来描述国民收入和政府支出的经济学模型。如果用$Y(t)$和$G(t)$分别表示国民收入和政府支出，那么凯恩斯模型可以表示为：

$$Y(t)=(1+a)\gamma Y(t-1)-a\gamma Y(t-2)+G(t) \tag{26-18}$$

其中a是加速因子，γ被称为动态因子。这是一个离散时间序列模型，下面考虑绘制这个模型的图形。这里取函数$G(t)$为下面的形式：

$$g(t)=2[t/4]+1 \tag{26-19}$$

其中[t/4]表示取不大于$t/4$的最大整数。

这里计算 3 种不同的初值情况，$a=0.92$，$\gamma=0.74$。相应的程序如下：

```
a=0.92; % 设置加速因子
g=0.74; % 设置动态因子
y1(1)=4;y1(2)=2; % 设置第一种初值情况
y2(1)=4;y2(2)=4; % 设置第二种初值情况
y3(1)=4;y3(2)=6; % 设置第三种初值情况
N=20; % 时间范围
for k=3:N;
    y1(k)=(1+a)*g*y1(k-1)-a*g*y1(k-2)+2*fix(k/4)+1; % 迭代计算
    y2(k)=(1+a)*g*y2(k-1)-a*g*y2(k-2)+2*fix(k/4)+1; % 迭代计算
    y3(k)=(1+a)*g*y3(k-1)-a*g*y3(k-2)+2*fix(k/4)+1; % 迭代计算
end
plot(1:N,y1,'kx',1:N,y2,'ko',1:N,y3,'ks'); % 绘图
set(gca,'Position',[0.2,0.2,0.6,0.4]); % 设置坐标轴位置
legend('C1','C2','C3',0); % 加注图例
xlabel('time'); % X 轴标注
ylabel('Y(t)'); % Y 轴标注
```

执行上述程序输出图形如图 26.2 所示。可见在不同的初始条件下，凯恩斯经济模型在一定时

间之后国民收入将不受初始时的国民收入影响。

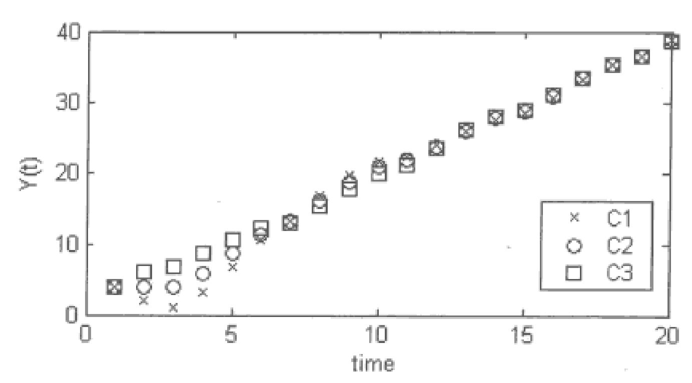


图 26.2 凯恩斯经济模型图形

在式（26-18）中，如果设 $x_1(t)=Y(t-2)$，$x_2(t)=Y(t-1)$，可以得到差分方程组：

$$\begin{cases} x_1(t+1)=x_2(t) \\ x_2(t+1)=-\gamma a x_1(t)+\gamma(1+a)x_2(t)+G(t) \end{cases} \tag{26-20}$$

上面的式子可以写为等价的矩阵方程形式，即：

$$\vec{x}(t+1)=\begin{bmatrix} 0 & 1 \\ -\gamma a & \gamma(1+a) \end{bmatrix}\vec{x}(t)+\begin{bmatrix} 0 \\ 1 \end{bmatrix}G(t) \tag{26-21}$$

其中 $\vec{x}=[x_1;x_2]$。式（26-21）的稳定性可以由下面 3 个不等式描述：

$$\begin{cases} \gamma a<1 \\ \gamma a>-1+\gamma(1+a) \\ \gamma a>-\gamma(1+a)-1 \end{cases} \tag{26-22}$$

如果 $a>0$ 且 $0<\gamma<1$，后两个不等式成立，因此再要求 $\gamma a<1$ 即可。式（26-21）的特征方程为：

$$\begin{vmatrix} \lambda & -1 \\ \gamma a & \lambda-\gamma(1+a) \end{vmatrix}=\lambda^2-\gamma(1+a)\lambda+\gamma a=0 \tag{26-23}$$

上式中关于 λ 的二次方程的判别式为：

$$\Delta=(1+a)^2\gamma^2-4\gamma a \tag{26-24}$$

取 $\Delta=0$，则有：

$$\begin{cases} \gamma=0 \\ \gamma=\dfrac{4a}{(1+a)^2} \end{cases} \tag{26-25}$$

利用下面的程序画出 $\gamma=1/a$ 和 $\gamma=4a/(1+a)^2$ 所对应的图形。

```
a=linspace(0,8,801); % 离散化参数 a
g=4*a./(1+a).^2;    % 计算参数 gamma 的数值
plot(a,g);hold on;   % 绘图
```

```
plot(a(a>0.5),1./a(a>0.5),'k'); % 画倒数
set(gca,'Position',[0.2,0.2,0.7,0.4]); % 设置坐标轴位置
xlabel('\ita','Fontsize',14,'Fontname','Times new roman'); % X 轴标注
ylabel('\it\gamma','Fontsize',14,'Fontname','Times new roman','Rotation',0); % Y
轴标注
text(1,1.6,'{\it\gamma} =1/{\ita}','Fontsize',14,'Fontname','Times new roman');
% 增加标注
text(3,0.85,'{\it\gamma} =4{\ita}/(1+{\ita})^2','Fontsize',14,'Fontname','Times
new roman'); %增加标注
```

执行上述程序所得图形如图 26.3 所示。

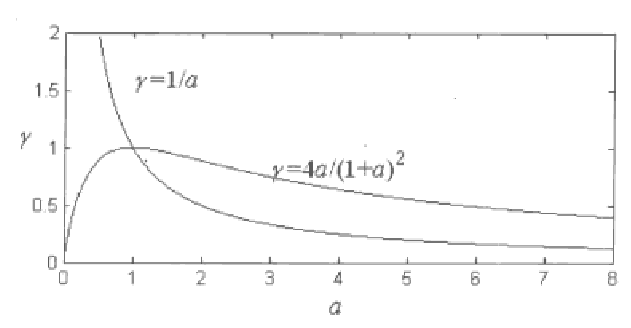


图 26.3　凯恩斯模型稳定性曲线

26.3.2　封闭经济系统的动态 IS-LM 模型

封闭经济系统的动态 IS-LM 模型用于描述商品市场和货币市场的动态变化规律，这个经济系统模型可以用一个微分方程表示：

$$\begin{bmatrix}\dot{y}\\ \dot{r}\end{bmatrix}=\begin{bmatrix}b(1-\tau)-1 & -d\\ k & -h\end{bmatrix}\begin{bmatrix}y\\ r\end{bmatrix}+\begin{bmatrix}a\\ -m_s\end{bmatrix} \tag{26-26}$$

其中 y 表示国民收入，r 表示利率。b，τ，d，k，h，a 和 m_s 为该模型系统的参数。

下面考虑这个微分方程系统的求解。取参数值为：$b=0.9$，$\tau=0.2$，$d=0.1$，$k=0.3$，$h=0.4$，$a=80$，$m_s=7$；初始条件为：$y(0)=2$，$r(0)=0.02$。

首先定义这个微分方程。

```
function dx=cislm(t,x);
% x(1)对应于 y
% x(2)对应于 r
b=0.9; % 对参数 b 赋值
tau=0.2; % 对参数 tau 赋值
d=0.1; % 对参数 d 赋值
k=0.3; % 对参数 k 赋值
h=0.4; % 对参数 h 赋值
a=80; % 对参数 a 赋值
ms=7; % 对参数 ms 赋值
dx=[b*(1-tau)-1,-d;k,-h]*x+[a;-ms]; % 定义微分方程
```

通过下面的程序调用函数 ode45 来求解这个微分方程：

```
tspan=[0,40]; % 定义时间范围
x0=[1,0.2]; % 定义初始条件
options=odeset('AbsTol',1e-8); % 设置微分方程求解选项参数
[t,y]=ode45('cislm',tspan,x0,options); % 求解微分方程
plot(t,y(:,1));hold on; % 绘制 t-y 曲线
plot(t,y(:,2),'r:'); % 绘制 t-r 曲线
xlabel('time','Fontsize',14); % X 轴标注
ylabel('{\ity}({\itt}) & {\itr}({\itt})','Fontsize',14); % Y 轴标注
L=legend('{\ity}({\itt})','{\itr}({\itt})',0); % 增加图例
set(L,'Fontsize',14); % 设置图例的字体大小
```

执行上述程序输出图形如图 26.4 所示。可见一定时间之后 y 和 r 趋于平稳状态，读者可以变化其他参数对这个模型进行研究。

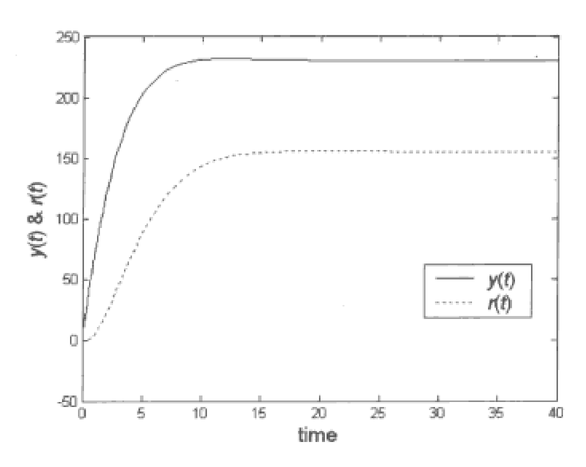


图 26.4 封闭经济系统的动态 IS–LM 模型

26.3.3 开放经济系统的动态 IS–LM–BP 模型

开放经济系统的动态 IS-LM-BP 模型引入国外经济动态情况，这样的系统模型变为开放型。该系统可以利用下面的矩阵方程表示：

$$\begin{bmatrix}\dot{y}\\ \dot{r}\\ \dot{s}\end{bmatrix}=\begin{bmatrix}m_{1,1} & m_{1,2} & m_{1,3}\\ m_{2,1} & m_{2,2} & 0\\ m_{3,1} & m_{3,2} & m_{3,3}\end{bmatrix}\begin{bmatrix}y\\ r\\ s\end{bmatrix}+\begin{bmatrix}c_1\\ c_2\\ c_3\end{bmatrix}=M\begin{bmatrix}y\\ r\\ s\end{bmatrix}+C \tag{26-27}$$

其中 y 表示国民收入，r 表示利率，s 表示国外经济状态。矩阵 M 为系数构成的矩阵。向量 C 是常数项。这是一个一阶三元微分方程。

下面考虑这个方程的求解。这里矩阵 M 的取值为：

$$M=\begin{bmatrix}-0.031 & -0.1 & 0.3\\ 0.2 & -0.4 & 0\\ -0.0002 & 0.001 & 0.0003\end{bmatrix} \tag{26-28}$$

向量 C 取值为 $C=[2.5 \quad -2 \quad -0.001]'$，初始条件为：$y(0)=40$，$r(0)=14$，$s(0)=2$。

首先定义微分方程组。

```
function dx=oislmbp(t,x);
% 对应关系：dx(1) -- y
%          dx(2) -- r
%          dx(3) -- s
M=[-0.031,-0.1,0.3;0.2,-0.4,0;-0.0002,0.001,0.0003]; % 计算参数矩阵 M
C=[2.5;-2;-0.001]; % 设置常数矩阵
dx=M*x+C; % 定义微分方程组
```

通过下面的程序调用函数 ode45 来求解这个微分方程。

```
tspan=[0,30]; % 设置求解时间的范围
x0=[40,14,2]; % 设置初始条件
options=odeset('AbsTol',1e-8); % 设置微分方程求解选项参数
[t,x]=ode45('oislmbp',tspan,x0,options); % 求解微分方程组
plot(t,x(:,1));hold on; % 绘制 t-y 曲线
plot(t,x(:,2),'r:');     % 绘制 t-r 曲线
plot(t,x(:,3),'k-.');    % 绘制 t-r 曲线
xlabel('time','Fontsize',14); % X 轴标注
ylabel('{\ity}({\itt}), {\itr}({\itt}) & {\its}({\itt})','Fontsize',14); % Y 轴
标注
L=legend('{\ity}({\itt})','{\itr}({\itt})','{\its}({\itt})',0); % 增加图例
ylim([0,50]); % 设置 Y 轴范围
set(L,'Fontsize',14); % 设置图例的字体大小
```

执行上述程序输出结果如图 26.5 所示，其中 $s(t)$ 稳定不变，而 $y(t)$ 和 $r(t)$ 随时间缓慢增加且增加速度越来越慢。

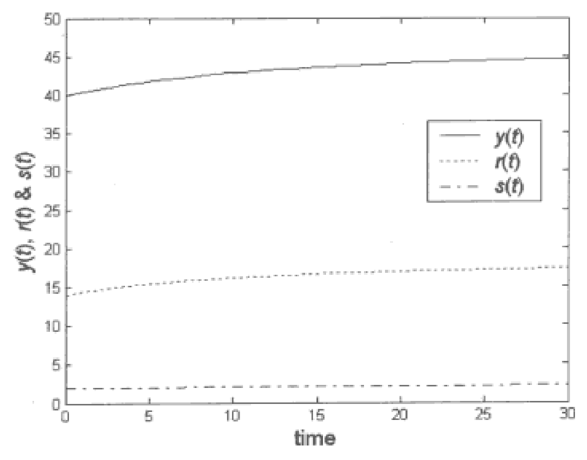


图 26.5　开放经济系统的动态 IS-LM-BP 模型曲线图

26.4　规划问题求解

在一些经济与管理相关问题中，经常遇到这样一类问题：在一些约束条件下求解最优化结果，使成本最低或者利润最大化。这样的问题可以使用前面介绍的线性规划函数来方便地处理，这类函数的用法已经在前面的章节中介绍过，本节直接举例说明具体问题的求解方法。

某公司在进行生产任务分配时，遇到这样一个问题：有 4 种商品需要交到 3 个不同的工厂进行生产，现在已经知道每个工厂生产成本、生产任务以及每种商品的需求量，它们的具体数值如表 26.7 所示。

表 26.7 一次规划中的成本、生产任务和需求量表格

工厂 \ 产品	p1	p2	p3	p4	任务合计
f1	6	3	9	8	9
f2	10	3	7	2	9
f3	2	6	2	9	8
需求量	8	4	5	9	

在表 26.7 中，f1，f2，f3 和 p1，p2，p3，p4 对应的 3×4 矩阵是成本表格，这里记为矩阵 C。而最右侧一列含有 3 个元素，它是各个工厂的任务分配指标，记为 C^{ol}；最下面一行含有 4 个元素，它是指市场上对于各个产品的需求量，记为 R^{ow}。在这样的条件下计算生产任务的分配和最小成本。

这个规划问题可以写为下面的优化模型，即：

$$\min F=\sum_{m=1}^{M}\sum_{n=1}^{N}c_{m,n}x_{m,n} \tag{26-29}$$

其中约束条件可以写为：

$$\begin{cases}\sum_{n=1}^{N}x_{m,n}=C_m^{ol},(m=1,2,\ldots,M)\\ \sum_{m=1}^{M}x_{m,n}=R_n^{ow},(n=1,2,\ldots,N)\\ x_{m,n}\ge0,,(m=1,2,\ldots,M,n=1,2,\ldots,N)\end{cases} \tag{26-30}$$

在函数 linprog 中要求输入的约束等式是线性方程组。而式（26-30）给出的约束条件相当于 $M+N$ 个线性方程组。本问题中的待求分配方案 x 是一个和矩阵 C 大小一样的矩阵，同时需要给出最小成本。在进行线性优化时，需要把矩阵 C 转化为一个向量。在 MATLAB 中，可以用 “C(:)” 把矩阵 C 转化为列向量，此外还可以使用函数 reshape 来重置矩阵的行数和列数。

下面建立求解优化问题的函数文件，相应的 MATLAB 程序内容如下：

```
function [x,F]=optimize(C,row,col);
% 求解优化问题的函数
B=[row(:);col(:)]; % 生成等式约束左侧的向量
[M,N]=size(C);  % 计算矩阵 C 的行数和列数
A=zeros(M+N,M*N); % 生成等式约束右侧的矩阵 A
for k=1:M;
   T=zeros(M,N); % 生成一个空矩阵
   T(k,:)=1;       % 对应行的元素设置为 1
   A(k,:)=[T(:)]';  % 把矩阵 T 转化为一个行向量
end
for k=1:N;
```

```
    T=zeros(M,N);   % 生成一个空矩阵
    T(:,k)=1;       % 对应列的元素设置为 1
    A(M+k,:)=[T(:)]';  % 把矩阵 T 转化为一个行向量
end
lb=zeros(M*N,1); % 设置下边界
options=optimset('MaxIter',1e6); % 设置最大迭代次数
[x,F]=linprog(C(:),[],[],A,B,lb,[],[],options); % 调用函数 linprog 进行线性优化
x=reshape(x,M,N); % 把向量 x 转化为矩阵
```

上述程序内容是一个函数文件，把这部分程序内容保存为 optimize.m 文件后，即可调用它求解带有等式约束的优化问题。这个函数的调用格式为：

```
[x,F]=optimize(C,row,col);
```

参数说明：其中 x 表示所得未知矩阵的数值。F 表示优化方案对应的最小值。

利用下面的语句可以实现表 26.7 描述的优化问题的结果。

```
C=[6,3,9,8;10,3,7,2;2,6,2,9]; % 输入成本矩阵的数据
row=[9,9,8]; % 输入列元素求和计算所得的向量，对应于各个工厂的生产任务
col=[8,4,5,9]; % 输入列元素求和计算所得的向量，对应于市场对每种商品的需求
[x,F]=optimize(C,row,col); % 调用 optimize 函数计算优化方案和最小值
x,F % 显示计算所得优化结果和最小值
```

上述程序运行后得到的结果如下：

```
x =    5.0000    4.0000    0.0000    0.0000
       0.0000    0.0000    0.0000    9.0000
       3.0000    0.0000    5.0000    0.0000
F =     76.0000
```

可见最小成本是 76，矩阵 x 给出了各个工厂生产每种产品的数量。

如果式（26-30）变为下面的形式：

$$\begin{cases} \sum_{n=1}^{N} x_{m,n} \le C_m^{ol}, (m=1,2,\dots,M) \\ \sum_{m=1}^{M} x_{m,n} \le R_n^{ow}, (n=1,2,\dots,N) \\ x_{m,n} \ge 0,, (m=1,2,\dots,M, n=1,2,\dots,N) \end{cases} \tag{26-31}$$

把语句“[x,F]=linprog(C(:),[],[],A,B,lb,[],[],options);”改为下面的形式：

```
[x,F]=linprog(C(:),A,B,[],[],lb,[],[],options);
```

即可求解含有不等式约束的优化问题。如果在约束条件中同时含有等式约束和不等式约束，可以利用函数 optimize 中的结构分别计算出等式约束和不等式约束中的系数矩阵和向量，然后利用函数 linprog 来计算优化问题即可。这里不再举例说明了。

26.5 小结

结合 MATLAB 强大的计算功能，在统计、优化以及相关的工具箱基础上建立了金融工具箱，可以针对不同经济学问题进行计算和相关的模拟。本章首先介绍了 MATLAB 提供的金融工具箱的基本函数功能。接下来介绍了几种产生时间序列数据的预测模型，给出了相关模型的数据处理程序，还介绍了几种经济数学模型，并给出相关模型的程序实现。最后介绍了 MATLAB 在求解优化问题的使用方法，利用线性优化相关函数来求解优化方案和最优数值。

◆ ◆ ◆

第 27 章　常用算法及 MATLAB 实现

本章包括

- **遗传算法**　介绍遗传算法原理及 MATLAB 提供的相关函数。
- **模拟退火算法**　介绍模拟退火算法的原理，并给出一维和二维最值问题的求解例子。
- **分步傅里叶算法**　介绍分步傅里叶算法的原理，同时给出孤子传播实例。
- **蚁群算法**　介绍蚁群算法原理，同时给出利用该算法求最短路径问题。
- **分水岭算法**　介绍分水岭算法的原理以及 MATLAB 提供的相关函数。
- **粒子群算法**　介绍粒子群算法的原理，同时给出利用粒子群算法求最大值的例子。
- **BP 算法**　介绍 BP 算法的原理以及 MATLAB 提供函数的用法。
- **最短路径 Dijkstra 和 floyd 算法**　介绍了这两个算法的基本原理。
- **3 个圆的外切圆算法**　介绍该算法的基本原理，同时给出一个计算示例。

随着科技的发展，学科发展不断细化，一些被频繁使用的计算方法被广大研究者称为算法。随后这些算法又被应用到不同的领域中。如此下去，随着人们对算法认识的不断深入，它将逐渐演变为一个模型或者工具。通过对算法的改进可以加强算法的功能或者扩大其应用范围。本章介绍一些常用算法的基本知识及对应的 MATLAB 程序实现，希望能够帮助读者快速了解这些常用算法。

27.1　遗传算法

遗传算法（Genetic algorithm，缩写为 GA）最初源于对生物系统所进行的计算机模拟研究，为此后人称之为遗传算法。生物的进化过程依靠染色体之间的交叉和变异现象来完成。通过对自然界中的生物遗传与进化机制的模仿，对于不同类型的科学问题，许多研究者设计了不同的编码方法来求问题的解。于是人们研究出不同的编码方式来模仿不同环境下生物体遗传性质，其中编码方法和遗传算法组成了不同类型的遗传算法。遗传算法是一种借助于生物的自然选择与自然遗传过程实现随机搜索的算法。遗传算法具有广泛性、整体搜索特性、无须借助辅助信息以及含启发式的随机搜索等特点。

个体或者生物体的编码可以利用二进制数据来表示，如：

```
x1=[1  0  1  0  1  1]
x2=[0  0  1  0  1  1  1  1]
```

通过对这些二进制数据进行计算实现遗传算法的演化过程，比如再生和变异等。

一个遗传算法的基本结构如图 27.1 所示。在遗传算法中重复着再生、交叉以及变异等现象。种群从任意初始状态开始，通过随机选择、交叉和变异环节，得到一群更适合虚拟环境的群体。从而使群体向搜索空间较好的区域进化。如此代代繁衍下去，最后收敛到最适应环境的一组群体，从而可以得到问题的最优解。

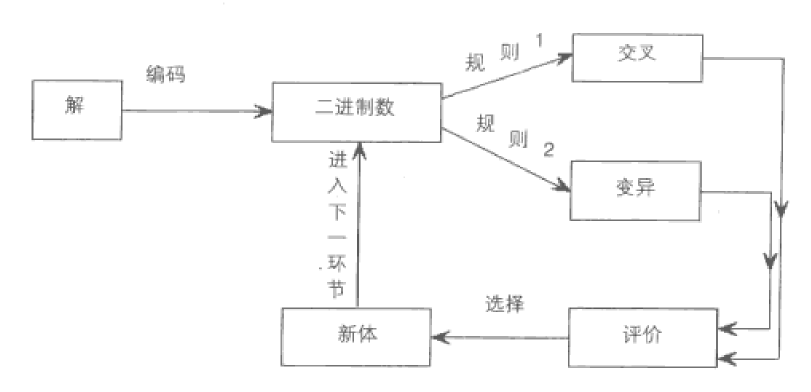


图 27.1 遗传算法的基本结构图

MATLAB 提供的遗传算法工具箱函数保存在 R2008a\toolbox\gads 文件夹下，其中关于遗传算法的函数如表 27.1 所示。

表 27.1 利用遗传算法求最小值的函数

函数名	功能说明
gaoptimset	设置遗传算法选项结构的属性值
gaoptimget	获取遗传算法选项结构的属性值
ga	利用遗传算法求单目标函数的最小值
gamultiobj	利用遗传算法求多目标函数的最小值

下面介绍表 27.1 中函数的用法。

1. 函数 gaoptimset

函数 gaoptimset 的调用格式为：

```
options = gaoptimset('param1',value1,'param2',value2,....);
```

参数说明： options 是输出的选项结构，其为结构体数据。param1 和 param2 表示属性名称。value1 和 value2 分别表示对应属性的取值。表 27.2 中给出了函数 gaoptimset 的属性说明及取值说明。

表 27.2 函数 gaoptimset 的属性参数说明

属性名	说明	取值说明
PopulationType	输入的人口类型	'bitstring' | 'custom' | {'doubleVector'}
PopInitRange	人口可能的初始取值范围	矩阵 | {[0;1]}
PopulationSize	表示个体数目的数值	正整数
EliteCount	不变化的下一代最优个体数目	正整数 | {2}
CrossoverFraction	个体间交换基因的比例	正数 | {0.8}
ParetoFraction	前面非受控人口比例	正数 | {0.35}
MigrationDirection	适宜个人流动的方向	'both' | {'forward'}
MigrationInterval	个体移民中代的数目	正整数 | {20}
Generations	允许代的最大数目	正整数 | {100}
TimeLimit	允许的最大时间	正整数 | {Inf}
FitnessLimit	期望的最小适应度函数	实数 | {-Inf}
StallGenLimit	小于 TolFun 的适应度函数值下累计改变的代的数目	正整数 | {50}

（续表）

属性名	说明	取值说明
StallTimeLimit	负数适应度函数值中改变的最大时间	正整数 | {20}
TolFun	适应度函数值的终止容限	正数 | {1e-6}
TolCon	约束的终止容限	正数 | {1e-6}
InitialPopulation	遗传算法中的初始人口	矩阵 | {[]}
InitialScores	用于确定适应度的初始评价	列向量 | {[]}
InitialPenalty	惩罚参数的初值	正数 | {10}
PenaltyFactor	罚函数更新参数	正数 | {100}
CreationFcn	产生初始人口的函数	@gacreationlinearfeasible | @gacreationuniform
FitnessScalingFcn	度量适应度的函数	@fitscalingshiftlinear | @fitscalingprop | @fitscalingtop | {@fitscalingrank}
SelectionFcn	用于下一代选择父辈的函数	@selectionremainder | @selectionuniform | @selectionroulette | @selectiontournament | @selectionstochunif
CrossoverFcn	执行交叉的函数	@crossoverheuristic | @crossoverintermediate | @crossoversinglepoint | @crossovertwopoint | @crossoverarithmetic | {@crossoverscattered}
MutationFcn	用于变异的函数	@mutationuniform | @mutationadaptfeasible | @mutationgaussian
DistanceMeasureFcn	测量个体间平均距离的函数	{@distancecrowding}
HybridFcn	备用优化函数，用于遗传算法终止情况	@fminsearch | @patternsearch | @fminunc | @fmincon | {[]}
Display	显示等级	'off' | 'iter' | 'diagnose' | {'final'}
OutputFcns	每一代中调用的函数，它比 PlotFcns 更普通	@gaoutputgen | {[]}
PlotFcns	画出模拟中各个量的函数	@gaplotbestf | @gaplotbestindiv | @gaplotdistance | @gaplotexpectation | @gaplotgenealogy | @gaplotselection | @gaplotrange | @gaplotscorediversity | @gaplotscores | @gaplotstopping | {[]}
PlotInterval	画图的代的数目	正整数 | {1}
Vectorized	向量化目标函数，同时在一次调用中可以评价多点	'on' | {'off'}
UseParallel	利用 PARFOR 评估目标函数和非线性约束函数	'always' | {'never'}



其中花括号表示默认值，@表示函数句柄，引号表示属性值为字符串。

2. 函数 gaoptimget

函数 gaoptimget 的调用格式为：

```
val = gaoptimget(options,'name');
```

参数说明： val 表示获取的属性值。options 是遗传算法选项结构。name 是属性名。

3. 函数 ga

函数 ga 的调用格式为：

```
x = ga(fitnessfcn,nvars);
x = ga(fitnessfcn,nvars,A,b);
x = ga(fitnessfcn,nvars,A,b,Aeq,beq);
x = ga(fitnessfcn,nvars,A,b,Aeq,beq,Lb,ub);
x = ga(fitnessfcn,nvars,A,b,Aeq,beq,Lb,ub,nonlcon);
x = ga(fitnessfcn,nvars,A,b,Aeq,beq,Lb,ub,nonlcon,options);
[x,fval] = ga(fitnessfcn, ...);
[x,fval,exitflag] = ga(fitnessfcn, ...);
[x,fval,exitflag,output] = ga(fitnessfcn, ...);
[x,fval,exitflag,output,population] = ga(fitnessfcn, ...);
[x,fval,exitflag,output,population,scores] = ga(fitnessfcn, ...);
```

参数说明： x 是适应度函数取最小值时的参数取值。fitnessfcn 表示适应度函数的句柄。nvars 是适应度函数的维数。A 和 b 分别是矩阵和向量，它们是不等式 $Ax \le b$ 的系数矩阵。Aeq 和 beq 分别是等式 Aeq*X=beq 中的系数矩阵和系数向量。Lb 和 ub 分别是变量的下界和上界。fval 是适应度函数的最小值。nonlcon 是描述非线性约束的函数句柄。exitflag 表示遗传算法存在条件，其可能取值及含义是：1 表示适应度函数值的平均变化在 StallGenLimit 属性值小于 TolFun 属性值并且约束违反小于 TolCon 范围外；3 表示适应度函数在属性限制范围内不变；4 表示步长小于机器精度同时约束违反小于 TolCon，这种情况只适用于非线性约束；5 表示适应度极限达到同时约束违反小于 TolCon；0 表示超过代的最大值；-1 表示输出或者 plot 函数终止优化；-2 表示找不到合适的点；-4 表示界限时间极限超出；-5 表示超出时间极限。output 表示输出的结构，其包含以下信息：randstate 表示随机函数 rand 的状态值；randnstate 表示随机函数 randn 的状态值；generations 表示总的代数，不包括混合迭代；funccount 表示函数计算总次数；maxconstraint 表示最大约束违反；message 给出遗传算法终止信息。population 是结束时的最终人口数目。scores 表示最后人口数目的评价。

这里给出利用函数 ga 计算下面二元函数最小值的例子。

$$f(x_1,x_2)=[4\cos(2x_1)+x_2]\exp(2x_2) \tag{27-1}$$

其中约束条件为：

$$x_1+x_2 \le 6\text{，}\quad 0 \le x_1, x_2 \le 6$$

分析： 这个问题是单个目标函数的最值问题，因此可以使用函数 ga 来计算。这里参数 nvars 等于 2，A 等于[1,2]，b 等于 6，上界 ub 等于 6，下界 Lb 等于 0。在这些已知参数下，考虑利用函数 ga 求最小值。

相应的程序如下：

```
fitness=inline('[4*cos(x(1)*2)+x(2)].*exp(2*x(2))'); % 定义适应度函数
A=[1,2]; % 定义不等式约束的系数矩阵
```

```
b=6; % 定义不等式约束的向量部分
Lb=[0,0]; % 定义下界
ub=[6,6]; % 定义上界
options=gaoptimset('TolFun',1e-4); % 定义遗传算法选项
[x,fval,exitflag] = ga(@(x)fitness(x),2,A,b,[],[],Lb,ub,[],options) % 利用函数ga
求最小值
```

上述程序输出结果如下：

```
x =    1.4900    2.2555
fval = -154.0240
exitflag =     1
```

这说明在 $x_1 = 1.4900$ 和 x_2=2.2555 时函数 $f(x_1, x_2)$取最小值-154.0240。

4. 函数 gamultiobj

函数 gamultiobj 的用法和函数 ga 相似，不同的是它可以计算多个函数的最小值。可以阅读这个函数的帮助信息。下面给出利用函数 gamultiobj 求多个目标函数最小值的例子。

两个目标函数如下定义：

$$f_1 = x_2 \cos x_1 + x_1 \sin x_2 \tag{27-2}$$

$$f2 = \sin(x_1 + x_2) + \cos(x_2 + x_3) + \sin(x_1 + x_3) \tag{27-3}$$

其中约束条件为 $-3 \le x_1, x_2, x_3 < 3$ 。计算 $[f_1, f_2]$ 的最小值。

分析：第一个函数中含有两个变量，而第二个函数中含有三个变量，所以参数 nvars 等于 3。这里约束条件中的不等式约束和等式约束条件都不存在，因此相应地在程序中可以使用空矩阵表示。而变量的上下界分别可以用[3,3,3]和[-3,-3,-3]表示。在此基础上考虑相应的程序实现。

相应的 MATLAB 程序如下：

```
f1= inline('x(2)*cos(x(1))+x(1)*sin(x(2))'); % 定义第一个函数
f2= @(x) sin(x(1)+x(2))+cos(x(2)+x(3))+sin(x(1)+x(3)); % 定义第二个函数
fun=@(x) [f1(x), f2(x)]; % 把两个函数组合为一个向量函数
Lb=[-3,-3,-3]; % 设置下界
ub=[3,3,3]; % 设置上界
options=gaoptimset('TolFun',1e-4); % 定义遗传算法选项
[x,fval] = gamultiobj(fun,3,[],[],[],[],Lb,ub); % 计算多目标函数的最小值
x,fval % 输出计算结果
```

上述程序部分输出结果如下：

```
x =    -2.9989    1.8951    1.3351
       -2.9988    1.7433    1.3529
       -2.9987    1.5256    1.5175
       -2.9987    1.5470    1.5037
```

限于篇幅，这里只给出这两个函数的使用方法的介绍。更多关于遗传算法方面的函数可以参阅相关书籍来深入了解。

27.2 模拟退火算法

模拟退火（Simulated annealing，缩写为 SA）算法是基于蒙特卡罗算法结构来求解的一种启

发式随机搜索算法，算法思想最早在 1953 年由 Metropolis 等人提出，而把该算法用于组合优化问题设计的是在 1983 年由 Kirkpatrick 等人和 Cerny 分别提出来的。这个算法把组合优化问题和统计学中的热平衡问题类比，开辟了求解组合优化问题的一个新途径。其出发点是取自物理学中退火过程的概念，即对固体物质进行退火处理时，通常是先将它加热到某一温度，使其内部粒子进行较大振幅的自由运动，然后降温操作，粒子逐渐形成较低能态的晶体。如果在凝结点附近温度下降得足够慢，那么固体物质一定会形成最低能量的状态。

模拟退火算法的基本思想是开始时给出一个试探解，然后从它的邻域中随机产生另一个解，这个新解要受到 Metropolis 提出的规则限制，这样目标函数在设定的范围内变化，这个变化过程由一个控制参数 t 决定， t 的作用类似于物理过程中的温度 T。对于控制参数 t 的每一个取值，模拟退火算法持续进行“产生—判断—接受（或舍去）”多次迭代计算，这个计算过程对应着固体在某一恒定温度下趋于热平衡的过程。在控制参数 t 逐渐减小并趋于 0 时，系统越来越趋于平衡态。最后所考虑的系统状态就对应于优化问题的全局最优解，这个过程也可以称为冷却过程。由于固体退火过程要求缓慢降温，这样才能使得固体在每一温度下都达到热平衡状态，而最终趋于平衡状态。因此控制参数 t 要通过缓慢的衰减，才能确保模拟退火算法最终优化问题的整体最优解。

图 27.2 给出了一般模拟退火算法的流程图。相应算法步骤可以总结如下：

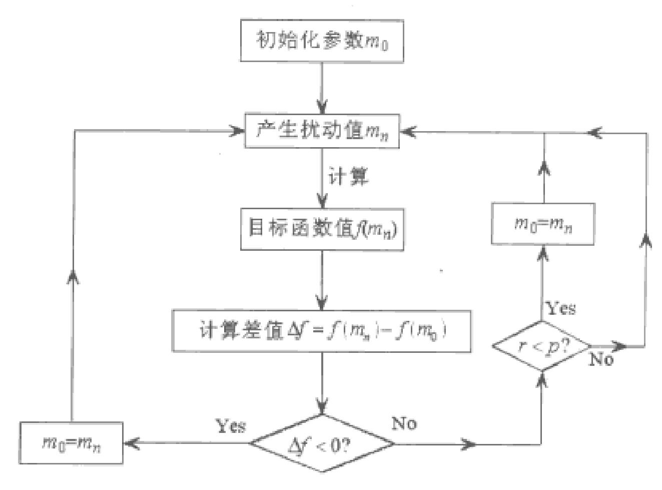


图 27.2 模拟退火算法的流程图

step 1 设置参数取值范围，给定一个初始的参数值 m_0，然后计算该值的目标函数取值 $f(m_0)$。

step 2 在 m_0 附近产生一个随机的扰动后得到一个新的参数值 m，同时把相应的目标函数值 $f(m)$ 计算出来。

step 3 计算出新旧参数对应的目标函数的差值 $\Delta f=f(m)-f(m_0)$。

step 4 如果 Δf 小于 0，则 $m_0=m$。若 Δf 大于 0，则 m_0 被更新（$m_0=m$）的几率为 p，其中 $p=\exp(-\Delta f/T)$，这里 T 表示温度值。

step 5 在同一温度 T 下，多次重复执行上面的步骤。

step 6 缓慢地降低温度 T，直至达到收敛条件为止。

实际上模拟退火算法包括两重循环，即参数的随机扰动和温度降低。该算法初始温度要求设计

在高温位置，这使得目标函数值增大的情况可能被捕捉到，因而可以舍弃局部的极小值。通过缓慢降低温度，算法最终可以收敛到全局范围上的最小点。此外，对于多元函数，参数 m 可以是一个向量形式。

这里对于模拟退火算法需要做以下几点说明：

- 当 $\Delta f<0$ 时，$\exp(-\Delta f/T)>1$，此时如果使用 0~1 之间的随机数 r，那么 $r<\exp(-\Delta f/T)$ 是成立的，因此可以使用判断条件 $r<p$ 来处理 $\Delta f<0$ 的情况。
- 对于温度的衰减这里使用指数衰减方式来控制，即 $T_n=T_0\lambda_T^n$，其中 T_0 是初始温度，T_n 表示第 n 个离散的温度数值，λ_T 是温度衰减指数因子。
- 对于扰动的最大步长，这里同样使用指数衰减的方式，即 $dm_n=dm_0\lambda_m^n$，其中 dm_n 表示第 n 步计算中使用的步长，dm_0 表示初始时的步长，λ_m 是步长衰减因子。

下面考虑利用模拟退火算法求一元函数 $f(m)$ 在区间 $[-3,3]$ 内最小值的问题。$f(m)$ 的表达式如下：

$$f(m)=\left(m-m^3-2\right)\exp\left(-m^2\right)\sin(4m) \tag{27-4}$$

计算中初始参数 $m_0=-0.5$，初始扰动步长 $dm_0=0.2$；初始温度 $T_0=8000$，每个温度值下的随机扰动次数为 100 次，$\lambda_m=0.99$，$\lambda_T=0.9$。相应的模拟退火计算程序如下：

```
set(gcf,'DoubleBuffer','on'); % 设置图形窗口的渲染效果
x=linspace(-3,3,201); % 离散采样点
plot(x,[x-x.^3-2].*exp(-x.^2).*sin(x*4),'k');hold on; % 绘制函数对应的曲线
m0=-0.5; % 设置初始参数值
dm=0.2;% 参数初始步长
fm0=[m0-m0^3-2]*exp(-m0^2)*sin(m0*4); % 计算相应的目标函数值
ph=plot(m0,fm0,'r+'); % 绘制当前位置对应的点
T=8000; % 设置初始温度
N=100; % 随机扰动的次数
Lm=0.99; % 设置参数步长衰减系数
LT=0.9; % 设置温度步长衰减系数
Tt=title(['T=',num2str(T)]); % 显示当前温度
rand('state',0); % 设置随机数的状态值
while T>0.001; % 循环模拟计算温度降低过程
    for k=1:100; % 多次进行随机扰动
        m=m0+dm*(2*rand-1); % 进行参数的随机扰动
        m=mod(m+3,6)-3; % 限制参数在考虑区间内变换
        fm=[m-m^3-2]*exp(-m^2)*sin(m*4); % 计算相应目标函数值
        Df=fm-fm0; % 计算新旧函数数值的差
        if rand<exp(-Df/T); % 以几率形式接受当前的参数值
            m0=m; % 更新 m0 的数值
            fm0=fm; % 更新 fm0 的数值
        end
        set(ph,'XData',m0,'YData',fm0); % 更新当前数据点的位置
        pause(0.05); % 暂停一下显示动画效果
    end
    T=T*LT; % 温度衰减
    dm=dm*Lm; % 参数步长衰减
    set(Tt,'String',['T=',num2str(T)]); % 更新当前温度
end
m, fm % 显示最小值位置和最小值
```



这个程序以动画的方式显示了模拟退火过程，执行中的一个界面如图 27.3 所示。最终十字标记停留在曲线的最低处。

输出结果为：

```
m =    0.3184
fm =   -1.4808
```

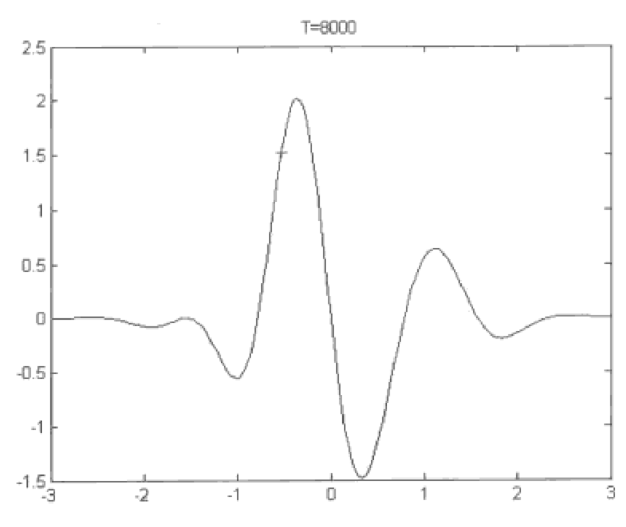


图 27.3　模拟退火算法计算一元函数的最小值

下面考虑利用模拟退火算法求解二元函数的最小值问题，这里以 MATLAB 函数 peaks 对应的函数为例求解其最小值。该函数的解析表达式为：

$$f(x,y)=3(1-x)^2\exp\left(-x^2-(y+1)^2\right)-10\left(\frac{x}{5}-x^3-y^5\right)\exp\left(-x^2-y^2\right)-\frac{1}{3}\exp\left(-(x+1)^2-y^2\right) \quad (27\text{-}5)$$

其中 $x,y\in[-3,3]$。

模拟退火算法的参数为 $x_0=y_0=0.5$，初始步长为 $dx_0=dy_0=0.2$，初始温度为 $T_0=8000$，每个温度下的随机扰动次数为 200，步长衰减系数为 $\lambda_m=0.99$，温度衰减系数为 $\lambda_T=0.85$。

相应的模拟退火算法的 MATLAB 程序为：

```
[X,Y,Z] = peaks(401); % 生成离散数据
x=X(1,:); % 提取 X 轴刻度
zx=min(Z,[],1); % 计算最小值在 X 轴投影的数据
y=Y(:,1); % 提取 Y 轴刻度
zy=min(Z,[],2); % 计算最小值在 Y 轴投影的数据
s(1)=subplot(121);plot(x,zx);axis([-3,3,-7,8.2]);hold on; % 绘图 z 在水平方向的最小
值曲线
Lh(1)=xlabel('\itx');Lh(2)=title('min({\itz}({\itx},{\ity}))_{\ity}'); % 标注 X
轴和图题
s(2)=subplot(122);plot(y,zy);axis([-3,3,-7,8.2]);hold on; % 绘图 z 在竖直方向的最小
值曲线
Lh(3)=xlabel('\ity');Lh(4)=title('min({\itz}({\itx},{\ity}))_{\itx}'); % 标注 X
轴和图题
```

```
ss=repmat(' ',1,16); % 16 个空格字符串
Lh(5)=ylabel(['T=8000',ss],'Rotation',0); % 显示当前温度
set(Lh,'Fontsize',14,'Fontname','Times New roman'); % 设置 Lh 句柄对象的字体属性
set(gcf,'Position',[78 75 1226 420],'DoubleBuffer','on'); % 设置图形窗口的位置和渲染效果
mx0=0.5; % X 坐标的初始值
my0=0.5; % Y 坐标的初始值
dm=0.2; % 参数初始步长
fm0=3*(1-mx0)^2*exp(-(mx0^2)-(my0+1)^2)-10*(mx0/5-mx0^3-my0^5)*exp(-mx0^2-my0^2)-1/3*exp(-(mx0+1)^2-my0^2); % 计算相应的目标函数值
ph1=plot(s(1),mx0,fm0,'r+'); % 绘制当前位置对应的点
ph2=plot(s(2),my0,fm0,'r+'); % 绘制当前位置对应的点
T=8000; % 设置初始温度
N=200; % 随机扰动的次数
Lm=0.99; % 设置参数步长衰减系数
LT=0.85; % 设置温度步长衰减系数
rand('state',0); % 设置随机数的状态值
while T>0.001; % 循环模拟计算温度降低过程
    for k=1:100; % 多次进行随机扰动
        mx=mx0+dm*(2*rand-1); % 进行参数的随机扰动
        mx=mod(mx+3,6)-3; % 限制参数 mx 在考虑区间内变换
        my=my0+dm*(2*rand-1); % 进行参数的随机扰动
        my=mod(my+3,6)-3; % 限制参数 my 在考虑区间内变换
fm=3*(1-mx)^2*exp(-(mx^2)-(my+1)^2)-10*(mx/5-mx^3-my^5)*exp(-mx^2-my^2)-1/3*exp(-(mx+1)^2-my^2);
% 计算相应目标函数值
        Df=fm-fm0; % 计算新旧函数值的差
        if rand<exp(-Df/T); % 以几率形式接受当前的参数值
            mx0=mx; % 更新 mx0 的数值
            my0=my; % 更新 mx0 的数值
            fm0=fm; % 更新 fm0 的数值
        end
        set(ph1,'XData',mx0,'YData',fm0); % 更新当前数据点的位置
        set(ph2,'XData',my0,'YData',fm0); % 更新当前数据点的位置
        pause(0.05); % 暂停一下显示动画效果
    end
    T=T*LT; % 温度衰减
    dm=dm*Lm; % 参数步长衰减
    set(Lh(5),'String',['T=',num2str(T),ss]); % 更新当前温度
end
mx,my,fm % 输出最小值位置和最小值
```

执行上述程序输出结果为：

```
mx =    0.1663
my =   -1.5603
fm =   -6.4338
```



为了比较实时点的位置和整体轮廓曲面的位置关系，这里绘制曲面在 XOZ 和 YOZ 平面的投影对应的最小值曲线，如图 27.4 所示。二者公共的最小值对应的 X 轴和 Y 轴的刻度就是最小值位置。当前点用十字符号表示，点应该在曲线上或者曲线的上部。同时在两个坐标轴之间实时地显示了当前的温度数值。

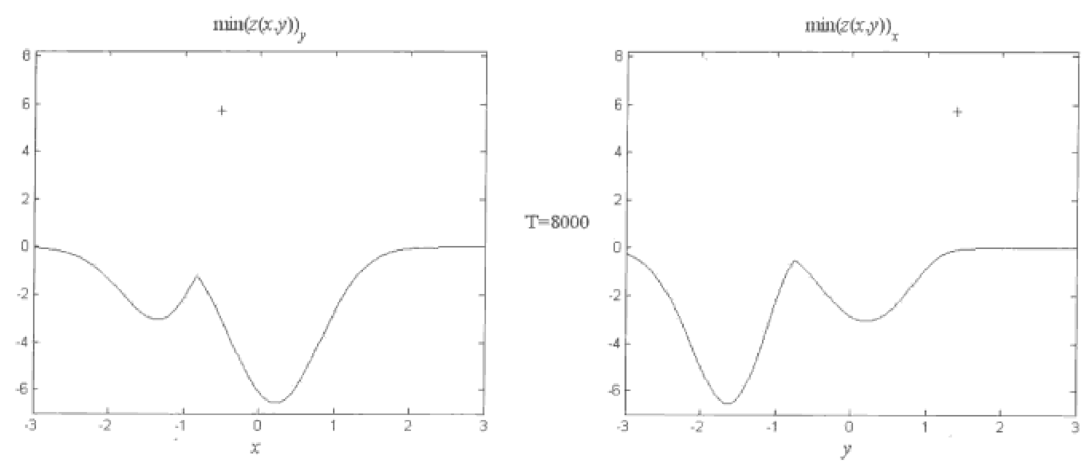


图 27.4 动态显示模拟退火算法的收敛过程

模拟退火是一种含随机数的算法，本身存在着不确定性。求得的优化值需要较大的计算量，甚至还有可能会陷入局部最优解，模拟退火算法的参数是根据经验选择的，合理地选取初始温度、降温方案等参数可以得到较好的结果。

27.3 分步傅里叶算法

分步傅里叶算法是在求解非线性薛定谔方程过程中发展起来的一种算法。下面先来介绍非线性薛定谔方程，该方程可以描述为：

$$\frac{\partial U}{\partial z}-\frac{i}{2}\frac{\partial^2 U}{\partial t^2}+(\alpha-\beta i)|U|^2 U=0 \tag{27-6}$$

其中 U 表示光场的复振幅，t 表示时间，α 和 β 为方程的系数。可以把等式（27-6）表示为下面的形式：

$$\frac{\partial U}{\partial z}=\left(\hat{D}+\hat{N}\right)U \tag{27-7}$$

其中，$\hat{D}$ 和 $\hat{N}$ 表示作用在 U 上的算法。它们的含义为：

$$\hat{D}U: i\frac{1}{2}\frac{\partial U}{\partial z}=-\frac{1}{2}\frac{\partial^2 U}{\partial t^2} \tag{27-8}$$

$$\hat{N}U: \frac{i}{2}\frac{\partial U}{\partial z}=-(i\alpha+\beta)|U|^2 U \tag{27-9}$$

这样在计算光从 z 到 $z+h$ 的传播时可以分解为两步。在 $[z,z+h/2]$ 时，公式（27-9）起作用，同时忽略 $|U|^2$ 在这个小区间内的变化，可以得到：

$$U(z+h/2,t)=U(z,t)\exp[i(\beta+i\alpha)|U|^2 h] \tag{27-10}$$

对等式（27-10）两端计算关于时间 t 的傅里叶变换，有：

$$U(z+h/2,\omega)=F\{\exp[i(\beta+i\alpha)|U|^2 h]U(z,t)\} \tag{27-11}$$

在式（27-11）中，ω 是傅里叶变换的频率。在后半部分的区间 $[z+h/2, z+h]$ 上，公式（27-8）起作用，同时对 U 关于时间 t 求傅里叶变换，可得：

$$F[U(z+h,t)] = U(z+h/2,\omega)\exp(-i\omega^2 h/2) \tag{27-12}$$

把式（27-11）代入式（27-12）并计算逆傅里叶变换，有：

$$U(z+h,t) = F^{-1}\{\exp(-i\omega^2 h/2)F[\exp(i(\beta+i\alpha)|U|^2 h)U(z,t)]\}$$

其中 F 和 F^{-1} 分别表示正、逆傅里叶变换。因为在 MATLAB 中可以使用函数 fft 和 iff 来计算正、逆傅里叶变换，这样可以比较快地求解这个微分方程。

在计算中，取相关参数为 $\alpha = 0.006$，$\beta = 1$，采样点数等于 200。相应的 MATLAB 程序如下：

```
a=0.006;  % 双光子吸收系数
b=1;      % 光学非线性系数
N=200;   % 设置采样点数
t=linspace(-20,20,N); % 取时间的离散值
z=linspace(0,50,N);  % 计算距离的离散值
h =z(2)-z(1);  % 计算距离步长
dt=t(2)-t(1);  % 计算时间步长
w=linspace(-1/dt/2,1/dt/2,N); % 计算傅里叶变换的频率值
Pw=exp(-i*w.^2*h/2*4*pi^2);  % exp(-i*omega^2*h/2)
% q1=zeros(N);q1(1,1:end)=sech(t);% 孤子的注入脉冲
U=zeros(N); % 初始化复振幅矩阵 U
U(1,1:end)=sech(t+3)+sech(t-3);% 设置孤子的注入脉冲作为初始条件
for k=2:N;
   ss2=exp(i*(b+i*a)*abs(U(k-1,:)).^2*h).*U(k-1,:); % 计算中间过程量
   S2=fftshift(fft(fftshift(ss2))); % 进行傅里叶变换
   U(k,1:end)=ifftshift(ifft(ifftshift(Pw.*S2))); % 进行逆傅里叶变换
end
mesh(t,z,abs(U)); % 显示计算结果
xlabel('\itt','fontsize',14); % X 轴标注
ylabel('\itz','fontsize',14); % Y 轴标注
zlabel('|U|','fontsize',14,'Rotation',0); % Z 轴标注
```

执行上述程序输出图形如图 27.5 所示，其中在 z=0 处是注入孤子双脉冲的形式，随着传播距离的增加双孤子出现一个较高的脉冲，然后二者之间的距离随着距离的增加而增加。

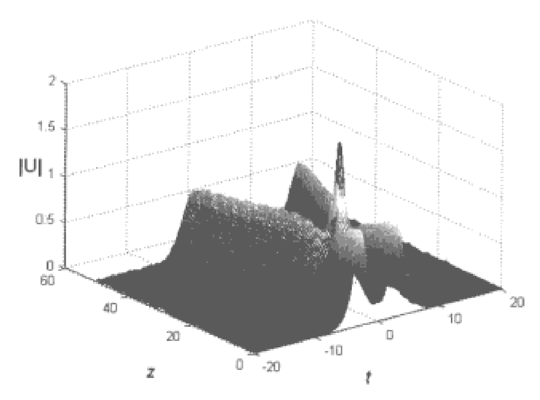


图 27.5 分步傅里叶算法计算的结果

27.4 蚁群算法

蚁群算法（Ant colony algorithm）最早是由意大利学者 Dorigo，Manierio 和 Collorni 等人提出的一种模拟进化算法。受到人们对现实蚁群集体行为研究成果的启发，由于蚁群搜索食物的过程与旅行商问题（Traveling salesman problem）非常相似，所以可以利用蚁群算法求解旅行商问题、指派问题（Assignment problem）和调度问题（Scheduling problem）。蚁群算法是一种适应性好、鲁棒性强，同时具有正反馈结构的并行算法。

对于自然界中的蚂蚁是如何找到从巢穴到食物源的最短路径问题，生物学家经过大量细致的观察研究后发现：最初单只蚂蚁行为是随机的。蚂蚁在运动过程中会在经过的路径留下一种信息物质。蚂蚁个体之间的信息传递就是通过这种物质进行的。一方面每只蚂蚁在走过的线路上留下一定量的信息物质，并且信息物质随时间衰减。另一方面其他蚂蚁能够感知这种信息物质并以路径上残留信息量指导行为，信息量越大，路径被选中的几率也越大。这样蚁群就可以快速地找到最佳路径。

可以在网上下载利用蚁群算法计算最短路径问题的程序（见附录 A），该程序由 GreenSim 团队于 2006 年初完成，网页上的程序是一个函数文件形式，保存后可以调用它，同时把其中的省会城市坐标数据保存为 gps.txt 文件，然后通过下面的语句调用这个函数文件：

```
m=31;Alpha=1;Beta=5;Rho=0.1;NC_max=200;Q=100; % 设置初始参数
C=textread('gps.txt',''); % 读入数据文件
[R_best,L_best,L_ave,Shortest_Route,Shortest_Length]=ACATSP(C,NC_max,m,Alpha,B
eta,Rho,Q); % 调用蚁群算法程序
```

输出图形如图 27.6 所示。

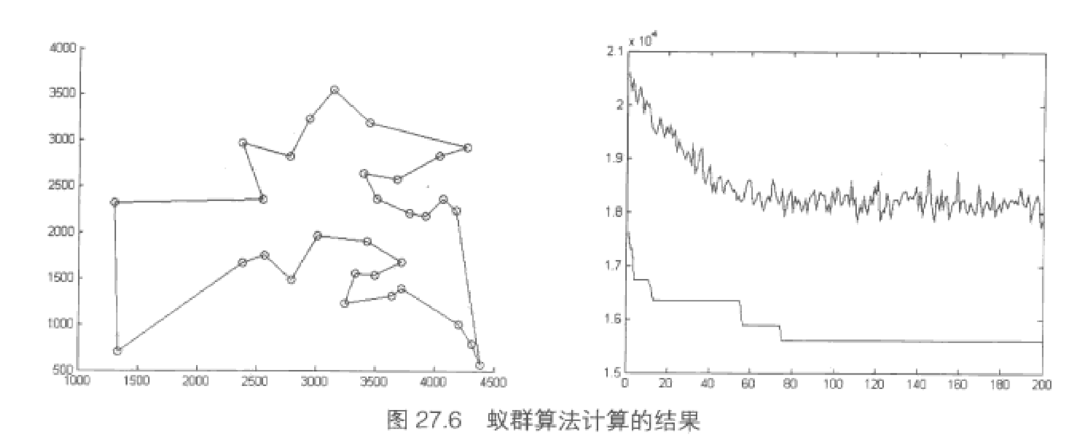


图 27.6 蚁群算法计算的结果

27.5 分水岭算法

分水岭算法是一种基于拓扑学中形态学的分割方法，其基本思想就是把图像看做是测地学上的拓扑地貌，而图像中每一点的像素值表示该点的海拔高度，每一个局部极小值及其相关区域被称为集水盆，集水盆的边界形成分水岭。分水岭的概念和形成可以使用模型浸入过程来说明。在每一个局部极小值附近刺穿一个孔，然后把整个模型慢慢浸入水中。随着浸入部分的增加，每一个局部极小值的影响域慢慢向四周扩展，在两个集水盆汇合处构筑大坝从而形成分水岭。分水岭的模拟过程

是一个迭代标注过程，而分水岭比较经典的算法是 Vincent 提出的。在 Vincent 算法中，分水岭计算分两个步骤：排序过程和淹没过程。首先对所有像素值进行从小到大排序，在从低到高进行淹没的过程中，对所有局部极小值在 h 高度的影响范围内采用“先进入先流出”的原则判断和标注。

MATLAB 提供的函数 watershed 可以实现分水岭变换，其调用格式为：

```
L = watershed(A);
L = watershed(A, conn);
```

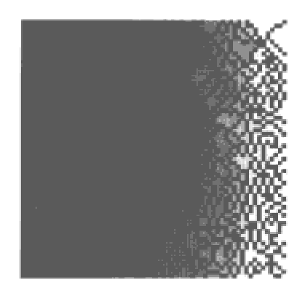
参数说明： L 是变换输出的图形。A 是输入图像对应的矩阵。conn 用于表示连通性，其可能取值如下：4 表示二维四近邻，8 表示二维八近邻，6 表示三维六近邻，18 表示三维十八近邻，26 表示三维二十六近邻。

下面举例说明函数 watershed 的用法。

```
A=imread('rice.png'); % 读入米粒图
A=A(1:64,1:64); % 取其 1/16 部分作为输入图像
L1=watershed(A,4); % 进行分水岭变换，conn=4
L2=watershed(A,8); % 进行分水岭变换，conn=8
subplot(131);imshow(A,[]);Xh(1)=xlabel('(a)'); % 绘制原图
subplot(132);imshow(L1,[]);Xh(2)=xlabel('(b)'); % 绘制分水岭变换的结果 1
subplot(133);imshow(L2,[]);Xh(3)=xlabel('(c)'); % 绘制分水岭变换的结果 2
set(Xh,'Fontsize',16,'Fontname','Times New Roman'); % 设置标注的字体大小
```

执行上述程序输出图像如图 27.7 所示，其中四近邻分水岭处理结果（图(b)）纹理较密，而八近邻分水岭处理结果（图(c)）纹理较为稀疏。


(a) 原图


(b) 变换结果 1

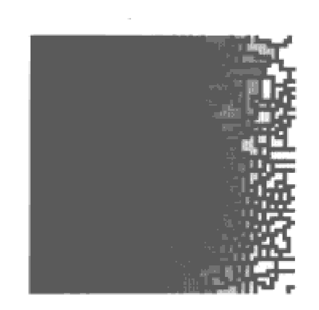

(c) 变换结果 2

图 27.7　分水岭变换的结果

27.6　粒子群优化算法

粒子群优化算法（Particle swarm optimization，缩写为 PSO）是一种群体智能算法，它是由美国心理学家 Kennedy 和电气工程师 Eberhart 在 1995 年首先提出来的。该算法的基本思想是对鸟群、鱼群在觅食过程中的迁徙和聚集行为的模拟，并利用生物学家 Heppner 的生物群体模型。粒子群优化算法是一类基于群体智能的随机优化技术，与遗传算法相比，二者都是基于群体的迭代搜索，但是粒子群优化算法没有交叉、变异操作，粒子群优化算法是通过个体之间的协作来搜寻最优解的。这个算法利用了生物群体中信息共享的思想，概念简单、易于实现，同时又有深刻的智能背景，适合科学研究和工程应用。粒子群优化算法的基本结构如图 27.8 所示。

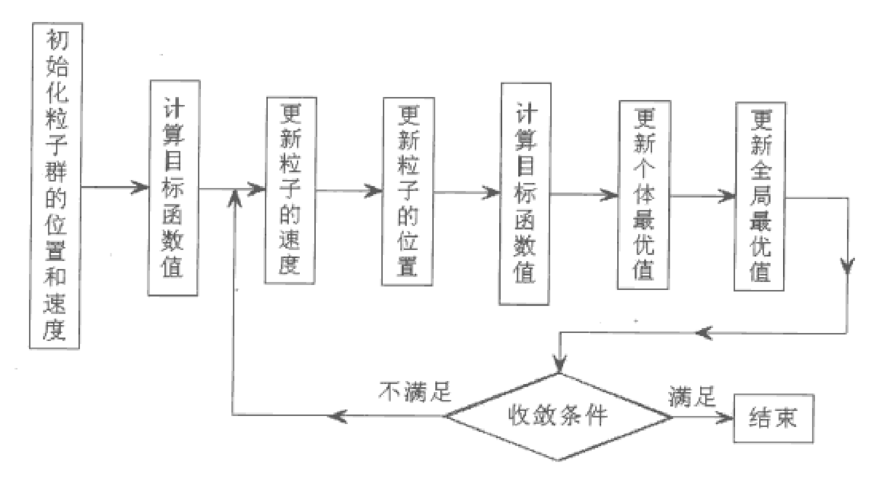


图 27.8 粒子群优化算法的基本原理图

在粒子群优化算法中，首先初始化一群随机粒子，然后通过迭代计算找到最优解。在每一次迭代中，粒子通过跟踪两个“极值”来更新自己：一个是粒子本身所找到的最优解，即个体极值；另一个是整个种群目前找到的最优解，即全局极值。在更新每个粒子位置的过程中，粒子最大速率被限制为 Vmax，粒子的坐标也被限制在所考虑的范围之内。其中粒子速度和位置的更新公式为：

$$V = wV + c_1R(P_{best} - P) + c_2R(G_{best} - P) \tag{27-13}$$

$$P = P + V \tag{27-14}$$

其中 V 是粒子的速度。w 是一个加权系数，其值一般取为 0.1~0.9 之间的数。P_{best} 是粒子自身最优位置。P 是粒子当前的位置。G_{best} 是粒子群全局最优位置。R 是一个 0~1 之间的随机数。在粒子群优化算法中，系数 c_1 和 c_2 被称为学习因子，它们一般取值为 2。

根据上面介绍的粒子群优化算法，利用粒子群优化算法计算下面函数的最大值，函数的定义区间为 $[0,9]$。

$$f(x) = \exp\left[-(x-4)^2\right]\cos(3x) \tag{27-15}$$

这里使用 8 个粒子来寻找函数的最大值，相应的 MATLAB 程序为：

```
N=8; %设置粒子总数
Vmax=0.2; % 最大速度
x=linspace(0,9,300); % 对变量 x 离散取值
y=exp(-[x-4].^2).*cos(x*3); % 计算相应的函数值
plot(x,y);hold on; % 绘制函数对应的曲线
c='kr'; % 设置表示颜色的字符串
Ls='sox+'; % 设置表示标记符号的控制符
Pn=linspace(1,8,N); % 设置初始粒子的位置
Fn=exp(-[Pn-4].^2).*cos(Pn*3); % 计算相应目标函数值
w=0.5; % 对权重系数赋值
c1=2; % 对学习因子赋值
c2=2; % 对学习因子赋值
V=Vmax*(2*rand(1,N)-1); % 初始化速度
Pbest=Pn; % 初始化个体最优位置 Pbest
Fbest=Fn; % 初始化最优目标函数值
[Mm,Ik]=max(Fn); % 找出最大值 Mm 和最大值的位置 Ik
Gbest=Pn(Ik(1)); % 得到当前全局最优位置 Gbest
```

```
Gfbest=max(Fn); % 目标函数的最大值
Gh=plot(Gbest*[1,1],[min(ylim),Gfbest],':'); % 利用虚线表示全局最大值位置
for n=1:N;
    mh(n)=plot(Pn(n),Fn(n),[c([n<4.5]+1),Ls(mod(n-1,4)+1)]); % 画出各个点
end
ti=title('time = 0','Fontsize',14,'Fontname','Times new roman');
for kk=1:150;
    R=rand(1,N); % 产生随机数
    V=w*V+c1*R.*(Pbest-Pn)+c2*R.*(Gbest-Pn); % 更新速度
    V(V>=Vmax)=Vmax; % 限制正方向最大速率
    V(V<=-Vmax)=-Vmax; % 限制负方向最大速率
    Pn=Pn+V; % 更新位置
    Fn=exp(-[Pn-4].^2).*cos(Pn*3); % 计算相应目标函数值
    Pbest(Fn>Fbest)=Pn(Fn>Fbest); % 更新最优目标位置
    Fbest(Fn>Fbest)=Fn(Fn>Fbest); % 更新目标函数值
    [Mm,Ik]=max(Fbest); % 从个体最大值中找到它们的最大值
    Ik(2:end)=[]; % 仅取 Ik 第一个数值
    Gfbest=max(Mm,Gfbest); % 更新全局最大值
    if numel(Ik)>0.5; % 若 Ik 是非空的
       Gbest=Pn(Ik); % 更新全局最大值
    end
    set(Gh,'XData',Gbest*[1,1],'YData',[min(ylim),Gfbest]); % 更新全局最大值对应的
虚线
    for k=1:N;
       set(mh(k),'XData',Pn(k),'YData',Fn(k)); % 更新各个点的位置
    end
    set(ti,'String',['time = ',num2str(kk)]); % 更新时间数值
    pause(0.05); % 暂停一下显示动画效果
end
Gbest,Gfbest % 显示最大值位置 Gbest 和最大值 Gfbest
```

执行上述程序输出的结果为：

```
Gbest =    4.1546
Gfbest =   0.9713
```

这里相应地给出了优化过程的动画演示，最终图形如图 27.9 所示。

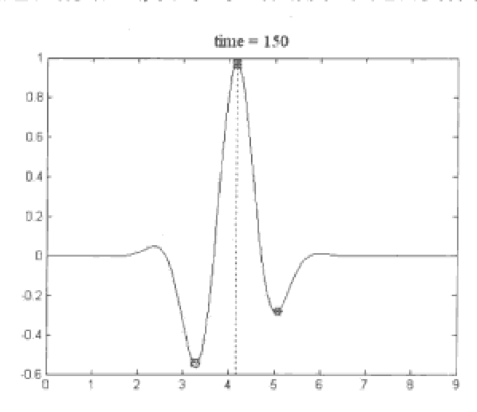


图 27.9　粒子群优化算法的动画演示界面图

粒子群优化算法在空间内的搜索过程中，有时会出现粒子在全局最优解附近“振荡”的现象。为了避免振荡现象，可以进行如下改进：在每步迭代中，速度 V 更新公式中的权重系数从最大加

权因子 w_max 线性减小到最小加权因子 w_min，即 w=w_max-iter* (w_max-w_min)/itermax，其中 iter 表示当前迭代的步数，而 itermax 是总迭代数。

27.7 BP 算法

人工神经网络（ANN）系统具有信息的分布存储和并行处理以及自学习能力等优点，同时它不需要对研究对象建模就能够较好地描述非线性系统和不确定性系统。目前这个系统已经在信息处理、控制以及系统建模等领域广泛应用，特别是基于误差反向传播算法的"多层前馈网络"算法（简称 BP 算法）能够以任意精度逼近连续函数，所以它广泛应用于非线性建模、逼近函数、模式识别和分类等问题中。为了解决 BP 算法收敛慢、训练时间长和目标函数存在局部最小等缺点，人们提出了一些改进算法。以神经网络为基础，MATLAB 提供了许多 BP 算法相关的函数，为人们利用 BP 算法进行研究提供了便利。几种常用的 BP 算法函数如表 27.3 所示。

表 27.3 常用 BP 算法函数

函数名	功能说明
trainrp	弹性 BP 算法，该函数只用导数符号表示权更新方向，不考虑导数大小，可以消除偏导数大小对权值的不利影响。具有收敛速度快、占内存小的特点
traingdx	自适应学习速率法，该算法检查权重修正值是否降低误差函数。如果降低，说明选取的学习速率有上升空间，可对其增加；反之，就减小学习速率。这样学习速率可根据误差性能函数调节，解决 BP 算法中学习速率选择不当的缺点
traincgf	共轭梯度法，其采用 Fletcher-Reeves 算法。该算法收敛速度较普通梯度下降法快很多。它需要线性搜索，存储量要求大。对于收敛速度，因问题而异。该算法计算代价较低，在较大规模问题中应用较广
trainbfg	拟牛顿算法，权值通过 x=x+a*dx 来修改，x 是搜索方向，a 是在 x 上的最小化性能函数。最初搜索方向沿梯度负方向，以后在迭代中按照 dx=-H-gx 来修改，H 为近似 Hessian 矩阵。该算法要求迭代次数较少，由于每步都要存储 Hessian 矩阵，故单步计算和存储量很大，它适合小型网络
trainlm	Levenberg-Marquardt 算法，权值通过 dx=-(JxT*jx+I*mu)-jxT*E 进行修正，其中 jx 是误差对权值微分的雅可比矩阵，E 是误差向量，mu 是调整量。该方法学习速度快，但占内存大，对于中等规模的网络来说是最好的一种训练算法。对于大型网络，可以通过设置参数 mem-reduc 把雅可比矩阵分成多个子矩阵，这样可减少内存消耗，但学习时间将会增大

表 27.3 中函数的调用格式大体相同，它们可以统一表示为：

```
[net, tr] = funname(NET, TR, trainV, valV, testV);
```

参数说明： net 表示训练过的神经网络。tr 是记录不同时期的数值。funname 表示表 27.3 所列的函数名。NET 是输入的神经网络。TR 是函数 train 生成的初始训练记录。trainV 是函数 train 生成的训练数据。valV 是函数 train 生成的确认数据。testV 是函数 train 生成的测试数据。

除了表 27.3 描述的函数外，MATLAB 还提供了函数 newff 来生成神经网络，其调用格式为：

```
net = newff(P,T,S,TF);
```

参数说明： net 是神经网络名称。P 是输入样本数据组成的矩阵。T 是每层神经元的个数。S 是 N-1 个隐藏层的 size。TF 是神经元传递函数。

除了表 27.3 列举的训练函数外，MATLAB 还提供了一个通用的训练函数，即 train，其调用格式为：

```
[net, tr] = train(NET, X, T, Pi, Ai);
```

参数说明： net 表示新的神经网络。tr 表示训练记录。NET 表示输入神经网络。X 表示神经网络的输入。T 是神经网络的目标。Pi 是初始输入延迟条件。Ai 是初始层延迟条件。这里 X，Pi 和 Ai 的默认值都是零。

下面举例说明 BP 算法函数的用法。

```
P=1:5; % 生成样本数据
T=[2,2,2,3,3]; % 生成神经元数据
net = newff(P,T,4,{'tansig','purelin'},'trainlm'); % 生成神经网络
net = train(net,P,T); % 进行训练
```

执行上面的程序可以弹出如图 27.10 所示的图形。单击 Performance 按钮将会弹出训练结果的曲线图，如图 27.11 所示。更多 BP 算法方面的函数可以参阅 MATLAB 帮助文档提供的信息。

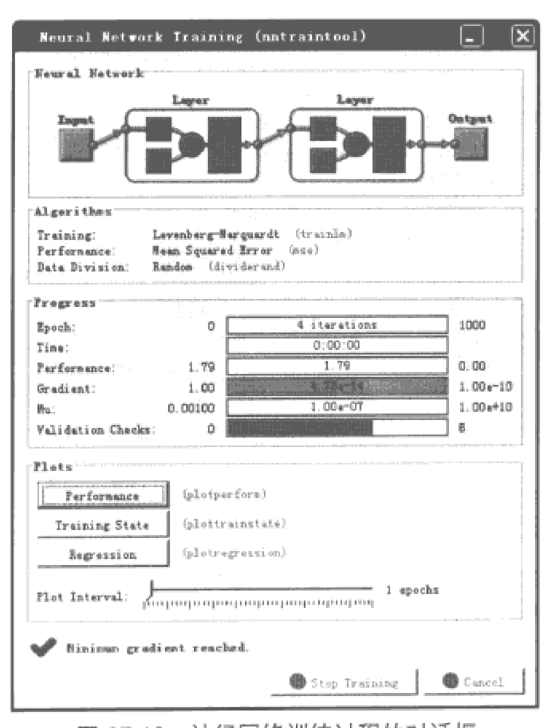


图 27.10　神经网络训练过程的对话框

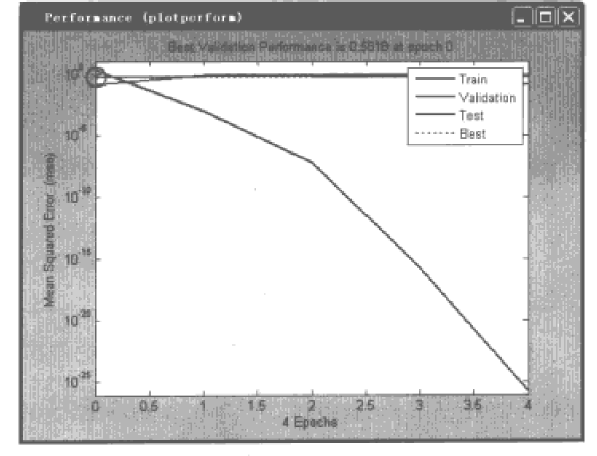


图 27.11　训练结果的曲线图形

27.8　最短路径 Dijkstra 和 floyd 算法

最短路径问题（Shortest-paths problem）作为图论中的一个经典问题，已经被应用于许多领域。在网络通信领域，信息包传递的路径选择问题与最短路径问题紧密相关。同时解决最短路径问题的相关算法在很多工程应用领域有较强的实用价值。本节来介绍相关的最短路径算法 Dijkstra 和 Floyd 算法。

Dijkstra 算法的基本思想是按距离从近到远为顺序，依次求得所有顶点的最短路径和距离，直至算法结束。为避免重复计算并保留每一步的计算信息，采用了标号算法。下面是该算法的计算步骤：

step 1 令 $L(u_0)=0$，如果 $v \neq u_0$，令 $L(v)=\infty$，$s_0=u_0$，$i=0$。

step 2 对所有 $v \in \overline{S}$（$\overline{S}=V \setminus S$），使用 $\min\limits_{u \in s_i}\left[L(v), L(u)+w(uv)\right]$ 代替 $L(v)$。计算 $\min\limits_{v \in s_i}\{L(v)\}$，把达到这个最小值的一个顶点记为 u_{i+1}，令 $S_{i+1}=S_i \cup \{U_{i+1}\}$。

step 3 若 $i=|v|-1$，计算停止；若 $i\langle|v|-1$，用 $i+1$ 代替 i，转至步骤 2。

算法结束时，从 u_0 到各顶点 v 的距离使用 v 最后一次的标号 $L(v)$ 给出。在 v 进入 S_i 之前的标号 $L(v)$ 记为 T 标号，v 进入 S_i 时的标号 $L(v)$ 记为 P 标号。该算法不断修改各顶点的 T 标号，直至获得 P 标号。若在算法运行过程中，将每一顶点获得 P 标号所由来的边在图上标明，则算法结束时，u_0 至各顶点的最短路径也在图上标示出来。

可以下载到 Dijkstra 算法的 MATLAB 程序（参见附录 A）。

Floyd 算法的基本思想是递推产生一系列 size 相同矩阵序列 $A_0, A_1, \cdots, A_k, \cdots,\ A_n$，其中矩阵 $A_k(i,j)$ 元素表示从顶点 v_i 到顶点 v_j 的路径上所经过顶点序号不大于 k 的最短路径长度。在计算时采用的迭代公式为：

$$A_k(i,j)=\min[A_{k-1}(i,j), A_{k-1}(i,k)+A_{k-1}(k,j)] \tag{27-16}$$

其中，k 表示迭代次数，$i,j,k=1,2,\ldots,n$。迭代终止时，A_n 就是各顶点之间的最短通路值。

可以从网上下载到 Floyd 算法的 MATLAB 程序（参见附录 A）。

27.9 3 个圆的外切圆算法

3 个圆的外切圆算法是作者在与 ivanhit 网友交流问题时考虑的一个算法。其算法原理如图 27.12 所示。

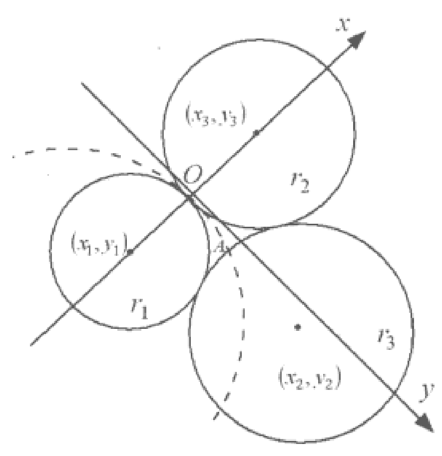


图 27.12 求 3 个圆的外切圆算法的原理图

算法的具体过程如下：首先计算出点 (x_1, y_1) 和点 (x_3, y_3) 的中点作为新坐标系的原点，这两点

的连线作为 X 轴，其法线方向取为 Y 轴。可以预见外接圆圆心应该在图 27.12 中 3 个圆围成的区域 A 内。这里设外接圆圆心坐标为(x,y)，半径为r，则有：

$$r+r_1=\sqrt{(x-x_1)^2+(y-y_1)^2} \tag{27-17}$$

$$r+r_3=\sqrt{(x-x_3)^2+(y-y_3)^2} \tag{27-18}$$

由式（27-17）和式（27-18）可知点(x,y)到点(x_1,y_1)和到点(x_3,y_3)的距离之差等于r_1-r_3，这是一个定值，那么点(x,y)应该在一条双曲线分支上，即图 27.12 中虚线表示的双曲线。唯一确定点(x,y)的坐标值的另外一个条件就是通过点(x_2,y_2)。同样地，点(x,y)到点(x_1,y_1)和到点(x_2,y_2)的距离之差等于r_1-r_2。在求解过程中根据双曲线可以确定点(x,y)的轨迹，利用点(x_2,y_2)确定点(x,y)坐标值时，可以使用二分法的思想多次反复计算即可得到最终坐标。此外利用递归过程可以得到多重外接圆的结果。上述过程可以通过光盘中\Ch27 文件夹下的 circles_1.m 文件实现。

通过下面的语句调用这个函数可以得到多重外接圆图案。

```
circles_1(5,-2,0,2,0,0,6);
```

执行上面的语句得到如图 27.13 所示的图形。

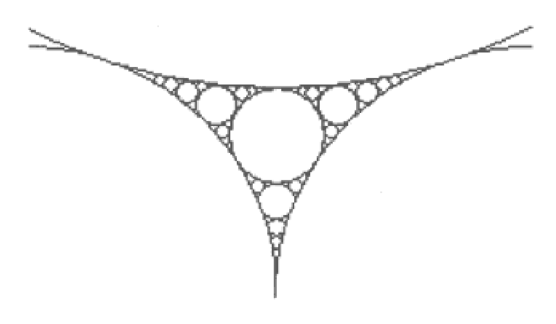


图 27.13　多重外接圆图案

27.10 小结

本章主要介绍了几种算法的 MATLAB 实现。算法在进行数值模拟研究时具有重要的作用，它是成熟数学模型的一个浓缩，研究者把它用通用的语言描述并用程序实现。这样不同领域的研究人员可以交叉使用这些算法，使得科学问题得到快速的求解。本章依次介绍了遗传算法、模拟退火算法、分步傅里叶算法、蚁群算法、分水岭算法、粒子群算法、BP 算法、最短路径求解算法以及 3 个点外接圆算法。通过介绍算法原理和程序实现，读者可以借鉴其中关键模块的程序实现方式，利用这些技巧在其他新型算法中应用。

◆　　◆　　◆

附录 A 网络程序下载地址

网络程序	下载地址
厄米多项式计算 MATLAB	http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=4911&objectType=FILE
勒让德多项式计算 MATLAB	http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=4710
LaguerrePoly.m 函数来计算拉盖尔多项式系数	http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=4912&objectType=File
多边形或者多角星为元素递归形成的分形图形	http://luobo.ycool.com/archive.56999.p5.html
分形盒维数值的计算程序用户	http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=13063&objectType=file
利用蚁群算法计算最短路径问题	http://blog.sina.com.cn/s/blog_4b425443010008re.html
Dijkstra 算法的 MATLAB 程序	http://www.mathworks.com/matlabcentral/fileexchange/5550， http://www.codesoso.com/Code/Dijkstra-Shortest-Path. aspx
Floyd 算法的 MATLAB 程序	http://www.mathworks.com/matlabcentral/fileexchange/11549

◆ ◆ ◆

参考文献

[1] 蒲俊，吉家峰，伊良忠．MATLAB6.0 数学手册．上海：浦东电子出版社，2002

[2] 许波，刘征．MatLab 工程数学应用．北京：清华大学出版社，2000

[3] 刘宏友，彭锋．MATLAB 6.x 符号运算及其应用．北京：机械工业出版社，2003

[4] 刘志俭．MATLAB 应用程序接口用户指南．北京：科学出版社，2000

[5] Eva Part-Enander，Anders Sjoberg 著． MATLAB5 手册．王艳清，孙锋，朱群雄等译．北京：机械工业出版社，2000

[6] 张志涌．精通 MATLAB 6.5 版教程．北京：北京航天航空大学出版社，2003

[7] 薛定宇，陈阳泉．高等应用数学问题的 MATLAB 求解．北京：清华大学出版社，2004

[8] 陆君安．偏微分方程的 MATLAB 解法．湖北：武汉大学出版社，2001

[9] 陆同兴．非线性物理概论．北京：中国科学技术大学出版社，2002

[10] 孙博文．分形算法与程序设计：Visual C++实现．北京：科学出版社，2004

[11] Bastien Chopard，Michel Droz 著．物理系统的元胞自动机模拟．祝玉学等译．北京：清华大学出版社，2003

[12] 于美文．光全息学及其应用．北京：北京理工大学出版社，1996

[13] 李工农，阮晓青，徐晨．经济预测与决策及其 Matlab 实现．北京：清华大学出版社，2007

[14] 王翼，王歆明．MATLAB 在动态经济学中的应用．北京：机械工业出版社，2006

[15] 葛哲学，孙志强．神经网络理论与 MATLAB R2007 实现．北京：电子工业出版社，2007

[16] 汪嘉业，杨兴强，张彩明．基于鱼眼镜头拍摄的图像生成漫游模型．系统仿真学报，2001，13 卷

[17] 方云团，沈廷根，谭锡林．双光子吸收效应对孤子和孤子对传输的影响．光学技术，2003，29 卷（5）：634~640

[18] Zhengjun Liu，Haifa Zhao，Shutian Liu．A discrete fractional random transform．Optics Communications，2005（255）：357~365